李炎恢 HTML CSS JavaScript PHP Bootstrap 教程

wizardforcel

Published with GitBook



目錄

李炎恢 HTML CSS JavaScript PHP Bootstrap 教程

HTML/CSS 教程

第 1 章 HTML5 概述

学习要点：

1.HTML5 的历史

2.HTML5 的功能

3.HTML5 的特点

4.课程学习问题

主讲教师：李炎恢

HTML5 是继 HTML4.01 和 XHTML1.0 之后的超文本标记语言的最新版本。 它是由一群自由思想者组成的团队设计出来，并最终实现多媒体支持、交互性、更加智能的表单，以及更好的语义化标记。

HTML5 并不仅仅是 HTML 规范的最新版本，而是一系列用来制作现代富 Web 内容的相关技术的总称，其中最重要的三项技术分别为：HTML5 核心规范（标签元素）、CSS（层叠样式表第三代）、和 JavaScript。

一 · **HTML5** 的历史

1993 年 HTML 首次以因特网草案的形式发布， 然后经历了 2.0、 3.2 和 4.0， 直到 1999年的 HTML4.01 版本稳定下来。由于发展缓慢，逐渐的被更加严格的 XHTML 取代。

XHTML 的兴衰史

自从 HTML4.01 版本之后，掌握着 HTML 规范的万维网联盟（W3C）组织没有再发布新的标准，而是围绕着 XHTML1.0 以及之后的 XHTML2.0 展开工作。XHTML 是基于 XML、致力于实现更加严格并且统一的编码规范的 HTML 版本，解决之前 HTML4.01 版本时，由于编码不规范导致浏览器的各种古怪行为。所以，Web 开发者对 XHTML 非常的拥护。XHTML 极大的好处，就是强迫开发者养成良好的编码习惯，放弃 HTML 的凌乱写法，最终降低了浏览器解析页面的难度，方便移植到更多平台。

可是，越是想往好的方面发展，往往可能是带来的却是毁灭性的灾难，世间万物就是如此。XHTML2.0 规范了更严格的错误处理规则，强制要求浏览器拒绝无效的 XHTML2 页面，强制 Web 开发者写出绝对正确规范的代码，同时不得向下兼容，摒弃 HTML 遗留的怪异行为和编码习惯。按理说，取其精华、舍其糟粕应该是好事。但是，这样的话，数亿的页面将无法兼容，Web 开发者的难度又被加大，并且制定这个标准又太过久远，最终被抛弃。

HTML5 的回归

2008 年 W3C 发布了 HTML5 的工作草案，2009 年停止了 XHTML2 计划。又过去大概一年， HTML5 规范进一步解决了诸多非常实际的问题， 各大浏览器厂商开始对旗下的产品进行升级，以便支持 HTML5。这样，得益于浏览器的实验反馈，HTML5 规范得到了持续的进步和完善，从而迅速融入到 Web 平台的实质性改进中。

和 XHTML2.0 不同， 制定 HTML5 规范的一群人并不想挑出以往 HTML 的各种毛病为其改正，而是尽可能的补全 Web 开发者急需的各种功能。这些功能包括更强大的 CSS3、表单验证、音频视频、本地存储、地理定位、绘画（Canvas）、Web 通信等等。

二 · **HTML5** 的功能

HTML5 到底涵盖了哪些功能？这些功能到底在主流的浏览器支持情况如何？

1.HTML5 核心：这部分主要由 W3C 官方的规范组成，涉及新的语义元素、新的增强的Web 表单、音频和视频、以及通过 JavaScript 绘图的 Canvas。这部分大多数主流浏览器均得到很好的支持；

2.曾经的 **HTML5** 标准：这部分主要来自于最初制定的 HTML5 规范，其中大多数功能需要 JavaScript 且支持富 Web 应用开发。比如：本地数据存储、离线应用和消息传递；3.非 HTML5 标准：这部分通常指下一代功能，虽然从未进入 HTML5 标准，但人们还是会把它认做 HTML5 的一部分。这些包括最为常见的 CSS3，以及很热门的地理定位。

对于最为常用且实用的部分， 基本上主流的浏览器都支持的比较好。 而那些特殊需求的部分，则需要根据不同的浏览器检测才能知道是否支持自己想要的功能。

三 · **HTML5** 的特点

在 HTML5 发展的同时，XHTML2.0 也在不断发展，那么到底是哪些特点导致 HTML5 取得最终的胜利呢？

1.向下兼容

对于 XHTML2.0 要求遵循规则，否则不予显示的方式，HTML5 却实行“不破坏 Web” 的原则。也就是说，以往已存在的 Web 页面，还可以保持正确的显示。

当然，面对开发者，HTML5 规范要求摒弃过去那些编码坏习惯和废弃的标签元素；而面对浏览器厂商，要求它们兼容 HTML 遗留的一切，以做到向下兼容。

2.用户至上

HTML5 遵循“用户至上”的原则，在出现具体问题时，会把用户放在第一位，其次是开发者，然后是浏览器厂商，最后才是规范制定者。比如，开发者在编码时不严谨导致本该出现警告或错误时，却正常显示了页面。

3.化繁为简

HTML5 对比之前的 XHTML，做了大量的简化工作。具体如下：

(1).以浏览器的原生能力代替复杂的 JavaScript；

(2).DOCTYPE 被简化到极致；

(3).字符集声明被简化；

(4).简单强大的 API。

4.无插件范式

在 HTML5 出现之前，很多功能只能通过插件或 hack（如绘图 API）来实现，但 HTML5原生提供了这些支持。使用插件有很多问题，具体如下：

(1).插件安装容易失败；

(2).插件被浏览器或软件禁用屏蔽（如 Flash 插件）；

(3).插件经常会被爆出漏洞被利用攻击；

(4).插件不容易与 HTML 文档其他部分集成（比如整体透明化等）。

5.访问通用性

这个原则分为三个概念：

(1).可访问性：比如更加利于残障人士的阅读方案；

(2).媒体中立：比如 HTML5 的媒体播放在不同设备或平台均能正常运行；

(3).支持所有语种：比如新元素<ruby>。

6.引入语义

HTML5 引入了一些用来区分不同含义和内容的标记元素。 这种方式极大的提供的编码人员的可读性和代码区域查询的便利性。

7.引入原生媒体支持

HTML5 的一次大改进救生衣支持在浏览器中直接播放视频和音频文件， 以前都需要借助插件才能实现此类功能。

8.引入可编程内容

HTML5 最大的变化就是引入了需要通过 JavaScript 编程才能完全的各种效果， 而这些很多都是 HTML5 原生的。那么现在 HTML5 可以理解为 HTML + CSS + JavaScript 的总称。

四 · 课程学习问题

学习 HTML5 需要一些测试用的浏览器、编码用的开发工具、以及建议推荐的学习方法。

1.浏览器选择

IE9+

Firefox 3.5+

Chrome 3.0+

Safari 3.0+

Opera 10.5+

这里重点要说明一下 IE 浏览器。由于历史和系统绑定原因，还有相当一部分电脑残留IE9 以下版本的浏览器。虽然微软已经开始发表声明逐步不再维护 IE8，但这部分群体还占有一定的份额。所以，是否要迎合这部分用户，取决于个人对市场的判断和成本的考量。

2.开发工具

本课程我们使用 Sublime Text3 作为 HTML5 课程的编码工具。使用了 Soda Dark 3作为软件界面的主题。

3.学习方式

本课程原则上是零基础、初学者可学，但如果你已经有 XHTML 课程基础，那么学习起来将非常轻松。当然，虽然我们已经录制了 XHTML 基础，在录制 HTML5 课程时，还是将所有学员当作刚接触的初学者来对待。 再当然， 这里所说的零基础和初学者也是需要一定经验，因为长期的教学工作，我们接触到很多连 QQ 不会用、邮件不会发送、迅雷不会下载的学员。这些学员是初学者之前的、负基础的学员，所以，如果是初学者一般问题不大。

第 2 章 基本格式

学习要点：

1.HTML5 文档结构

2.文档结构解析

3.元素标签探讨

主讲教师：李炎恢

本章主要先从 HTML5 的文档结构谈起。 这些基础元素确定了 HTML 文档的轮廓以及浏览器的初始环境。几乎所有页面都必须首先键入这些元素。

一 · **HTML5** 文档结构

1.第一步：打开 Sublime Text 3，打开指定文件夹；

2.第二步：保存 index.html 文件到磁盘中，.html 是网页后缀；

3.第三步：开始编写 HTML5 的基本格式。

```
<!DOCTYPE html> //文档类型声明 <html lang="zh-cn"> //表示 HTML 文档开始 <head> //包含文档元数据
```

二 · 文档结构解析

1.Doctype

文档类型声明（Document Type Declaration，也称 Doctype）。它主要告诉浏览器所查看的文件类型。 在以往的 HTML4.01 和 XHTML1.0 中， 它表示具体的 HTML 版本和风格。而如今 HTML5 不需要表示版本和风格了。

```
<!DOCTYPE html> //不分区大小写
```

2.html 元素

首先，元素就是标签的意思，html 元素即 html 标签。html 元素是文档开始和结尾的元素。它是一个双标签，头尾呼应，包含内容。这个元素有一个属性和值：lang="zh-cn"，表示文档采用语言为：简体中文。

```
<html lang="zh-cn"> //如果是英文则为 en
```

3.head 元素

用来包含元数据内容，元数据包括：<link>、<meta>、<noscript>、<script>、<style>、<title>。这些内容用来浏览器提供信息，比如 link 提供 CSS 信息，script提供 JavaScript 信息，title 提供页面标题等。

```
<head>...</head> //这些信息在页面不可见
```

4.meta 元素

这个元素用来提供关于文档的信息，起始结构有一个属性为：charset="utf8"。表示告诉浏览器页面采用的什么编码，一般来说我们就用 utf8。当然，文件保存的时候也是utf8，而浏览器也设置 utf8 即可正确显示中文。

```
<meta charset="utf-8"> //除了设置编码，还有别的
```

5.title 元素

这个元素很简单，顾名思义：设置浏览器左上角的标题。

```
<title>基本结构</title>
```

6.body 元素

用来包含文档内容的元素，也就是浏览器可见区域部分。所有的可见内容，都应该在这个元素内部进行添加。

```
<body>...</body>
```

7.a 元素

一个超链接，后面会详细探讨。

```
<a href="http://www.baidu.com">百度</a>
```

三．元素标签探讨

HTML 是一种标记语言，刚才的结构我们已经详细探讨过。这里，我们再剖析一下这些“标记”或者叫“标签”，书面上经常称作为“元素”的东西是怎么构成的。

1.元素

元素就是一组告诉浏览器如何处理一些内容的标签。每个元素都有一个关键字，比如<body>、<title>、<meta>都是元素。不同的标签名称代表不同的意义，后面将会涉及到段落标签、文本标签、链接标签、图片标签等。

元素一般分为两种：单标签（空元素）和双标签。单标签一般用于声明或者插入某个元素，比如声明字符编码就用<meta>，插入图片就用<img>；双标签一般用于设置一段区域的内容，将其包含起来，比如段落<p>...</p>。

2.属性和值

元素除了有单双之分，元素的内部还可以设置属性和值。这些属性值用来改变元素某些方面的行为。比如超链接：<a>中的 href 属性，里面替换网址即可链接到不同的网站。当然一个元素里面可以设置多个属性，甚至自定义属性。

第 3 章 文本元素

学习要点：

1.文本元素总汇

2.文本元素解析

主讲教师：李炎恢

本章主要探讨 HTML5 中的文本元素，所谓文本元素，就是将一段文本设置成相匹配的结构和含义。

一．文本元素总汇

HTML5 规范指出：使用元素应该完全从元素的语义出发。但是由于历史遗留及用户至上的原则，这种语义会宽松许多。

元素名称	说明
a	生成超链接
br	强制换行
wbr	可安全换行
b	标记一段文字但不强调
strong	表示重要
i	表示外文或科学术语
em	表示强调
code	表示计算机代码
var	表示程序输出
samp	表示变量
kdb	表示用户输入
abbr	表示缩写
cite	表示其他作品的标题
del	表示被删除的文字
s	表示文字已不再确认
dfn	表示术语定义
mark	表示与另一段上下文有关的内容
q	表示引自他处的内容
rp	与 ruby 元素结合使用，标记括号
rt	与 ruby 元素结合使用，标记括号
ruby	表示位于表意文字上方或右方的注音符号
bdo	控制文字的方向
small	表示小号字体内容
sub	表示下标字体
sup	表示上标字体
time	表示时间或日期
u	标记一段文字但不强调
span	通用元素，没有任何语义。一般配合 CSS 修饰。

从上面这张表格中，我们发现文本元素还是非常多的。但实际上，在现实应用中，真正常用的也就是那么几个，绝大部分是针对英文的。

二．文本元素解析

1.<b>表示关键字和产品名称

```
<b>HTML5</b>
```

解释：<b>元素实际作用就是加粗。从语义上来看，就是标记一段文字，但并不是特别强调或重要性。比如：一段文本中的某些关键字或者产品的名称。

2.<strong>表示重要的文字

```
<strong>HTML5</strong>
```

解释：<strong>元素实际作用和<b>一样，就是加粗。从语义上来看，就是强调一段重要的文本。

3.
强制换行、<wbr>安全换行

```
<br> Thisabc<wbr>dkedkslakdj<wbr>fkdlsakd is apple.
```

解释：在任意文本位置键入
都会被换行，而在英文单词过长时使用<wbr>会根据浏览器的宽度适当的裁切换行。

4.<i>表示外文词汇或科技术语

```
<i>HTML5</i>
```

解释：<i>元素实际作用就是倾斜。从语义上来看，表示区分周围内容，并不是特别强调或重要性。

5.<em>加以强调

```
<em>HTML5</em>
```

解释：<em>元素实际作用和<i>一样，就是倾斜；从语义上来看，表示对一段文本的强调。

6.<s>表示不准确或校正

```
<s>HTML5</s>
```

解释：<s>元素实际作用就是删除线；从语义上来看，表示不准确的删除。

7.<del>表示删除文字

```
<del>HTML5</del>
```

解释：<del>元素实际作用和<s>一样，就是删除线；从语义上来看，表示删除相关文字。

8.<u>表示给文字加上下划线

```
<u>HTML5</u>
```

解释：<u>元素实际作用就是加一条下划线；从语义上来看，并不强调此段文本。

9.<ins>添加一段文本

```
<ins>HTML5</ins>
```

解释：<ins>元素实际作用和<u>一样，加一条下划线；从语义上来看，是添加一段文本，起到强调的作用。

10.<small>添加小号字体

```
<small>HTML5</small>
```

解释：<small>元素实际作用就是将文本放小一号。从语义上来看，用于免责声明和澄清声明。

11.<sub><sup>添加上标和下标

```
<sub>5</sub>

<sup>5</sup>
```

解释：<sub>和<sup>元素实际作用就是数学的上标和下标。语义也是如此。

12.<code>等表示输入和输出

```
<code>HTML5</code>

<var>HTML5</var>

<samp>HTML5</samp>

<kdb>HTML5</kdb>
```

解释：<code>表示计算机代码片段；<var>表示编程语言中的变量；<samp>表示程序或计算机的输出；<kdb>表示用户的输入。由于这属于英文范畴的，必须将 lang="en"英语才能体现效果。

13.<abbr>表示缩写

```
<abbr>HTML5</abbr>
```

解释：<abbr>元素没有实际作用；从语义上看，是一段文本的缩写。

14.<dfn>表示定义术语

```
<dfn>HTML5</dfn>
```

解释：<dfn>元素就是一般性的倾斜；从语义上看，表示解释一个词或短语的一段文本。

15.<q>引用来自他处的内容

```
<q>HTML5</q>
```

解释：<q>元素实际作用就是加了一对双引号。从语义上来看，表示引用来自其他地方的内容。

16.<cite>引用其他作品的标题

```
<cite>HTML5</cite>
```

解释：<cite>元素实际作用就是加粗。从语义上来看，表示引用其他作品的标题。

17.<ruby>语言元素

```
<ruby> 饕<rp>(</rp><rt>tāo</rt><rp>)</rp> 餮<rp>(</rp><rt>tiè</rt><rp>)</rp> </ruby>
```

解释：<ruby>用来为非西方语言提供支持。<rp><rt>用来帮助读者掌握表意语言文字正确发音。比如：汉语拼音在文字的上方。但目前 Firefox 还不支持此特性。

18.<bdo>设置文字方向

```
<bdo dir="rtl">HTML5</bdo>
```

解释：<bdo>必须使用属性 dir 才可以设置，一共两个值：rtl（从右到左）和 ltr（从左到右）。一般默认是 ltr。还有一个<bdi>元素也是处理方向的，由于是特殊语言的特殊效果，且主流浏览器大半不支持，忽略。

19.<mark>突出显示文本

```
<mark>HTML5</mark>
```

解释：<mark>实际作用就是加上一个黄色的背景，黑色的字；从语义上来看，与上下文相关而突出的文本，用于记号。

20.<time>表示日期和时间

```
<time>2015-10-10</time>
```

解释：<time>元素没有实际作用；从语义上来看，表示日期和时间。

21.<span>表示一般性文本

```
<span>HTML5</span>
```

解释：<span>元素没有实际作用；语义上就是表示一段文本，我们经常用它来设置 CSS 等操作。

第 4 章 超链接和路径

学习要点：

1.超链接的属性

2.相对与绝对路径

3.锚点设置

主讲教师：李炎恢

本章主要探讨 HTML5 中文本元素最重要的一个超链接，探讨它自身的属性以及路径问题。

一．超链接的属性

<a>元素属于文本元素，有一些私有属性或者叫局部属性。那么，相对应的还有通用属性或叫做全局属性。这方面的知识，后面会详细探讨。

属性名称	说明
href	指定<a>元素所指资源的 URL
hreflang	指向的链接资源所使用的语言
media	说明所链接资源用于哪种设备
rel	说明文档与所链接资源的关系类型
target	指定用以打开所链接资源的浏览环境
type	说明所链接资源的 MIME 类型（比如 text/html）

在这几个属性当中，只有 href 和 target 一般比较常用，而 href 是必须要用的。其他几个属性，在<a>元素使用较少，将在 CSS 章节再探讨。

1.href 属性

```
<a href="http://www.baidu.com">百度</a>
```

解释：href 是必须属性，否则<a>元素就变成空元素了。如果属性值是 http://开头的 URL，意味着点击跳转到指定的外部网站。

2.target 属性

```
<a href="http://www.baidu.com" target="_blank">百度</a>
```

解释：target 属性告诉浏览器希望将所链接的资源显示在哪里。

属性名称	说明
_blank	在新窗口或标签页中打开文档
_parent	在父窗框组（frameset）中打开文档
_self	在当前窗口打开文档（默认）
_top	在顶层窗口打开文档

这四种最常用的是_blank，新建一个窗口。而_self 是默认，当前窗口打开。_parent和_top 是基于框架页面的，分别表示在父窗口打开和在整个窗口打开。而 HTML5 中，框架基本被废弃，只能使用<iframe>元素，且以后大量结合 JavaScript 和 PHP 等语言配合，框架用的就很少了。

二 · 相对与绝对路径

所谓相对路径，就是相对于链接页面而言的另一个页面的路径。而绝对路径，就是直接从 file:///磁盘符开始的完整路径。我们在同一个目录下做上两个页面，其中一个页面链接到另一个页面。

1.绝对路径

```
<a href="file:///D:/备课/HTML5 第一季/code/index2.html">index2</a>
```

解释：首先是 file:///开头，然后是磁盘符，然后是一个个的目录层次，找到相应文件。这种方式最致命的问题是，当整个目录转移到另外的盘符或其他电脑时，目录结构一旦出现任何变化，链接当即失效。

2.相对路径

```
<a href="index2.html">index2</a>
```

解释：相对路径的条件是必须文件都在一个磁盘或目录下，如果是在同一个目录下，直接属性值就是被链接的文件名.后缀名。如果在同一个主目录下，有多个子目录层次，那就需要使用目录结构语法。

3.目录语法

同一个目录：index2.html 或./index2.html；

在子目录：xxx/index2.html；

在孙子目录：xxx/xxx/index2.html；

在父目录：../index2.html；

在爷爷目录：../../index2.html；

三·锚点设置

超链接也可用来将同一个文档中的另一个元素移入视野。通过属性 id 或 name 实现锚点定位。

//链接

```
<a href="#1">第一章</a>
<a href="#2">第二章</a>
<a href="#3">第三章</a> //锚点
&lt;a name="1"&gt;&lt;/a&gt;&lt;a id="3"&gt;&lt;/a&gt;
```

第 5 章 分组元素

学习要点：

1.分组元素总汇

2.分组元素解析

主讲教师：李炎恢

本章主要探讨 HTML5 中分组元素的用法。所谓分组，就是用来组织相关内容的 HTML5 元素，清晰有效的进行归类。

一．分组元素总汇为了页面的排版需要，**HTML5** 提供了几种语义的分组元素。

元素名称	说明
p	表示段落
div	一个没有任何语义的通用元素，和 span 是对应元素
blockquote	表示引自他出的大段内容
pre	表示其格式应被保留的内容
hr	表示段落级别的主题转换，即水平线
ul,ol	表示无序列表，有序列表
li	用于 ul,ol 元素中的列表项
dl,dt,dd	表示包含一系列术语和定义说明的列表。dt 在 dl 内部表示术语，一般充当标题；dd 在 dl 内部表示定义，一般是内容。
figure	表示图片
figcaption	表示 figure 元素的标题

二．分组元素解析

1.<p>建立段落

```
<p> 这是一个段落 </p>
<p> 这也是一个段落 </p>
```

解释：<p>元素实际作用就是将内部包含的文本形成一个段落；而段落和段落之间保持一定量的空隙。

2.<div>通用分组

```
<div> 这是一个通用分组 </div>
<div> 这是又一个通用分组 </div>
```

解释：<div>元素在早期的版本中非常常用，通过<div>这种一般性分组元素进行布局。

而在 HTML5 中，由于语义的缘故，被其他各种文档元素所代替。和<p>段落的区别就是，两段文本的上下空隙是没有的，空隙间隔和
换行一样。

3.<blockquote>引用大段他出内容

```
<blockquote> 这是一个大段引自他出内容 </blockquote>
<blockquote> 这是另一个大段引自他出内容 </blockquote>
```

解释：<blockquote>元素实际作用除了和<p>元素一样，有段落空隙的功能，还包含了首尾缩进的功能。语义上表示，大段的引用他出的内容。

4.<pre>展现格式化内容

```
<pre> #####

        #####

            #####

        #####

    ##### </pre>
```

解释：<pre>元素实际作用就是编辑器怎么排版的，原封不动的展现出来。当然，这种只适合简单的排版，复杂的排版就无法满足要求了。

5.<hr>添加分隔

```
<hr>
```

解释：<hr>元素实际作用就是添加一条分割线，意图呈现上下文主题的分割。

6.
添加无序列表

```
<ul>
    <li> 张三 </li>
    <li> 李四 </li>
    <li> 王五 </li>
    <li> 马六 </li>
</ul>
```

解释：<ul>元素表示无序列表，而<li>元素则是内部的列表项。

7.
添加有序列表

```
<ol>
    <li> 张三 </li>
    <li> 李四 </li>
    <li> 王五 </li>
    <li> 马六 </li>
</ol>
```

解释：<ol>元素表示有序列表，而<li>元素则是内部的列表项。<ol>元素目前支持三种属性

ol 元素属性

属性名称	说明
start	从第几个序列开始统计：<ol start="2">
reversed	是否倒序排列：<ol reversed>，一半主流浏览器不支持
type	表示列表的编号类型，值分别为：1、a、A、i、I

li 元素属性

属性名称	说明
value	强行设置某个项目的编号。

```
<li value="7">王五</li>
```

8.<dl><dt><dd>生成说明列表

```
<dl>
    <dt> 这是一份文件 </dt>
    <dd> 这里是这份文件的详细内容 1 </dd>
    <dd> 这里是这份文件的详细内容 2 </dd>
</dl>
```

解释：这三个元素是一个整体，但<dt>或<dd>并非都必须出现。

9.<figure><figcaption>使用插图

```
<figure>
    <figcaption> 这是一张图 </figcaption>
    <img src="img.png">
</figure>
```

解释：这两个元素一般用于图片的布局。

第 6 章 表格元素

学习要点：

1.表格元素总汇

2.构建表格解析

主讲教师：李炎恢

本章主要探讨 HTML5 中表格元素的用法。表格的主要用途是以网格的形式显示二维数据。

一．表格元素总汇

表格的基本构成最少需要三个元素：<table>、<tr>、<td>，其他的一些作为可选辅助存在。

元素名称	说明
table	表示表格
thead	表示标题行
tbody	表示表格主体
tfoot	表示表脚
tr	表示一行单元格
th	表示标题行单元格
td	表示单元格
col	表示一列
colgroup	表示一组列
caption	表示表格标题

二．构建表格解析

1.<table><tr><td>构建基础表格

```
<table border="1">
    <tr>
        <td>张三</td>
        <td>男</td>
        <td>未婚</td>
    </tr>
    <tr>
        <td>李四</td>
        <td>女</td>
        <td>已婚</td>
    </tr>
</table>
```

解释：<table>元素表示一个表格的声明，<tr>元素表示表格的一行，<td>元素表示一个单元格。默认情况下表格是没有边框的，所以，在<table>元素增加一个 border 属性，设置为 1 即可显示边框。

2.<th>为表格添加标题单元格

```
<table border="1" style="width:300px;">
    <tr>
        <th>姓名</th>
        <th>性别</th>
        <th>婚姻</th>
    </tr>
    <tr>
        <td>张三</td>
        <td>男</td>
        <td>未婚</td>
    </tr>
    <tr>
        <td>李四</td>
        <td>女</td>
        <td>已婚</td>
    </tr>
</table>
```

解释：<th>元素主要是添加标题行的单元格，实际作用就是将内部文字居中且加粗。这里使用了一个通用属性 style，主要用于 CSS 样式设置，以后会涉及到。<th><td>均属于单元格，包含两个合并属性：colspan、rowspan 等。

3.<thead>添加表头

```
<thead>
    <tr>
        <th>姓名</th>
        <th>性别</th>
        <th>婚姻</th>
    </tr>
</thead>
```

解释：<thead>元素就是限制和规范了表格的表头部分。尤其是以后动态生成表头，它的位置是不固定的，使用此元素可以限定在开头位置。

4.<tfoot>添加表脚

```
<tfoot>
    <tr>
        <td colspan="3">统计：共两名</td>
    </tr>
</tfoot>
```

解释：<tfoot>元素为表格生成表脚，限制在表格的底部。

5.<tbody>添加表主体

```
<tbody>
    <tr>
        <td>张三</td>
        <td>男</td>
        <td>未婚</td>
    </tr>
    <tr>
        <td>李四</td>
        <td>女</td>
        <td>已婚</td>
    </tr>
</tbody>
```

解释：<tbody>元素主要是包含住非表头表脚的主体部分，有助于表格格式的清晰，更加有助于后续 CSS 和 JavaScript 的控制。

6.<caption>添加表格标题

```
<caption>这是一个人物表</caption>
```

解释：<caption>元素给表格添加一个标题。

7.<colgroup>设置列

```
<colgroup span="2" style="background:red;">
```

解释：<colgroup>元素是为了处理某个列，span 属性定义处理哪些列。1 表示第一列， 2 表示前两列。如果要单独设置第二列，那么需要声明两个，先处理第一个，将列点移入第二位，再处理第二个即可。

8.<col>更灵活的设置列

```
<colgroup>
    <col>
    <col style="background:red;" span="1">
</colgroup>
```

解释：<col>元素表示单独一列，一个表示一列，控制更加灵活。如果设置了 span 则，控制多列。

第 7 章 文档元素

学习要点：

1.文档元素总汇

2.文档元素解析

主讲教师：李炎恢

本章主要探讨 HTML5 中文档元素，文档元素的主要作用是划分各个不同的内容，让整个布局清晰明快。让整个布局元素具有语义，进一步替代 div。

一．文档元素总汇文档元素基本没有什么实际作用效果，主要目的是在页面布局时区分各个主题和概念。

元素名称	说明
h1~h6	表示标题
header	表示首部
footer	表示尾部
nav	表示有意集中在一起的导航元素
section	表示重要概念或主题
article	表示一段独立的内容
address	表示文档或 article 的联系信息
aside	表示与周边内容少有牵涉的内容
hgroup	将一组标题组织在一起
details	生成一个区域，用户将其展开可以获得更多细节
summary	用在 details 元素中，表示该元素内容的标题或说明

二．文档元素解析

文档元素的大部分标签，是没有任何效果的，完全是为了配合语义使用，以强调它的结构性。只有在后面的章节学习 CSS，配合使用才能起到布局和样式的效果。

1.<header>表示首部

```
<header>      这里部分一般是页面头部，包括：LOGO、标题、导航等内容 </header>
```

解释：<header>元素主要设置页面的标头部分。

2.<footer>表示尾部

```
<footer> 这里是页面的尾部，一般包括：版权声明、友情链接等内容 </footer>
```

解释：<footer>元素主要设置页面的尾部。

3.<h1>~<h6>添加标题

```
<h1>标题部分</h1>

&lt;h4&gt;小标题部分&lt;/h4&gt;
```

解释：<h1>~<h6>实际作用就是加粗并改变字体大小。用于各种标题文档。

4.<hgroup>组合标题

```
<hgroup>
    <h1>标题部分</h1>
    <h4>小标题部分</h4>
</hgroup>
```

解释：<hgroup>元素的作用就是当多个标题出现，干扰到一对或多个本身需要整合的标题，这是使用此元素包含群组。

5.<section>文档主题

```
<section> 这里一般是存放文档主题内容。 </section>
```

解释：<section>元素的作用是定义一个文档的主题内容。

6.<nav>添加导航

```
<nav>这里存放文档的导航</nav>
```

解释：<nav>元素给文档页面添加一个导航。

7.<article>添加一个独立成篇的文档

```
<article>
    <header>
        <nav></nav>
    </header>
    <section></section>
    <footer></footer>
</article>
```

解释：<article>元素表示独立成篇的文档，里面可以包含头、尾、主题等一系列内容。在比较大的页面中会使用到，比如一片博文的列表，每篇博文，都有自己的头、尾、主题等内容。和此相似的还有论坛的帖子、用户的评论、新闻等。

8.<aside>生成注释栏

```
<aside>这是是一个注释</aside>
```

解释：<aside>元素专门为某一段内容进行注释使用。

9.<address>表示文档或 **article** 元素的联系信息。

```
<address>联系信息</address>
```

解释：如果是在<body>元素下时，表示整个文档的联系信息。如果是在<article>元素下时，表示其下的联系信息。

10.<details>元素生成详情区域、**<summary>**元素在其内部生成说明标签

解释：由于大多数主流浏览器尚未支持，暂略。

第 8 章 嵌入元素

学习要点：

1.嵌入元素总汇

2.嵌入元素解析

主讲教师：李炎恢

本章主要探讨 HTML5 中嵌入元素，嵌入元素主要功能是把外部的一些资源插入到HTML 中。

一．嵌入元素总汇

这里所列出的元素，并非本节课全部涉及到，比如音频 audio、视频 video、以及动态图像 canvas 和媒体资源 source、track 等会在后面章节或季度讲解。

元素名称	说明
img	嵌入图片
map	定义客户端分区响应图
area	表示一个用户客户端分区响应图的区域
audio	表示一个音频资源
video	表示一个视频资源
iframe	嵌入一个文档
embed	用插件在 HTML 中嵌入内容
canvas	生成一个动态的图形画布
meter	嵌入数值在许可值范围背景中的图形表示
object	在 HTML 文档中嵌入内容
param	表示将通过 object 元素传递给插件的参数
progress	嵌入目标进展或任务完成情况的图形表示
source	表示媒体资源
svg	表示结构化矢量内容
track	表示媒体的附加轨道（例如字幕）

二．嵌入元素解析

1.<img>嵌入图像**

```
<img src="img.png">
```

解释：<img>元素主要是插入一张外部的图片，那么图片的路径问题和超链接一致。

img 的私有属性

属性名称	说明
src	嵌入图像的 URL
alt	当图片不加载时显示的备用内容
width	定义图片的长度（单位是像素）
height	定义图片的高度（单位是像素）
ismap	创建服务器端分区响应图
usemap	关联<map>元素

```
<a href="index.html">
<img src="img.png" width="339" height="229" alt="风景图" ismap> </a>
```

2.<map>创建分区响应图**

```
<img src="img.png" alt="风景图" width="339" height="229" usemap="#Map"> <map name="Map">
<area shape="rect" coords="31,28,112,100" href="http://www.baidu.com" target="_blank" alt
<area shape="circle" coords="187,142,47" href="http://www.google.com" target="_blank" alt
<area shape="poly" coords="287,26,240,66,308,112" href="http://www.soso.com" target="_bla
</map>
```

解释：通过图片中的热点进行超链接，这里我们采用 Dreamweaver 进行操作生成的。

3.<iframe>嵌入另一个文档**

```
<a href="index.html" target="in">index</a> | <a href="http://www.baidu.com" target="in">百

&lt;iframe src="http://www.ycku.com" width="600" height="500" name="in"&gt;&lt;/iframe&gt
```

解释：<iframe>表示内嵌一个 HTML 文档。其下的 src 属性表示初始化时显示的页面， width 和 height 表示内嵌文档的长度和高度，name 表示用于 target 的名称。

4.<embed>嵌入插件内容**

```
<embed src="http://www.tudou.com/v/i4ZZvFwfluI/&bid=05&rpid=50797543&resourceId=50797543_
```

解释：<embed>元素是扩展浏览器的功能，大部分用于添加对插件的支持。这里添加了一个土豆网的 flash 视频。type 类型表示被插入内容的类型，这里不列出所有，后面如果遇到其他类型的文件，再继续探讨；width 和 height 表示宽高；src 表示文件路径。

5.<object>和**<param>**元素**

解释：<object>元素和<embed>一样，只不过 object 是 html4 的标准，而 embed 是 html5 的标准。而 object 不但可以嵌入 flash，还可以嵌入图片等其他内容。由于图片、音频、视频、插件都有相应标签元素代替，我们这里暂时不对其详细讲解。

6.<progress>显示进度**

```
<progress value="30" max="100"></progress>
```

解释：<progress>元素可以显示一个进度条，一般通过 JS 控制内部的值。IE9 以及更低版本不支持此元素。

7.<meter>显示范围里的值**

```
<meter value="90" min="10" max="100" low="40" high="80" optimum="60"></meter>
```

解释：<meter>元素显示某个范围内的值。其下的属性包含：min 和 max 表示范围边界， low 表示小于它的值过低，high 表示大于它的值过高，optimum 表示最佳值，但不出现效果。IE 浏览器不支持此元素。

第 9 章 音频和视频

学习要点：

1.音频和视频概述

2.video 视频元素

3.audio 音频元素

主讲教师：李炎恢

本章主要探讨 HTML5 中音频和视频元素，通过这两个原生的媒体元素向 HTML 页面中嵌入音频和视频。

一．音频和视频概述

首先，我们要理解两个概念：容器（container）和编解码器(codec)。

1.视频容器****

音频文件或视频文件，都只是一个容器文件。视频文件包含了音频轨道、视频轨道和其他一些元数据。视频播放时，音频轨道和视频轨道是绑定在一起的。元数据包含了视频的封面、标题、子标题、字幕等相关信息。主流视频容器支持的格式为：.avi、.flv、.mp4、.mkv、.ogg、.webm。

2.**编解码器**

音频和视频编码/解码是一组算法，用来对一段特定音频或视频进行解码和编码，以便音频和视频能够播放。原始的媒体文件体积非常巨大，如果不对其进行编码，那么数据量是非常惊人的，在互联网上传播则要耗费无法忍受的时间；如果不对其进行解码，就无法将编码后的数据重组为原始的媒体数据。主流的音频编解码器：AAC、MPEG-3、Ogg Voribs，视频编解码器：H.264、VP8、Ogg Theora。

3.浏览器支持情况****

起初，HTML5 规范本来打算指定编解码器，但实施起来极为困难。部分厂商已有自己的标准，不愿实现标准；而有一些编解码器受专利保护，需要支付昂贵费用。最终放弃了统一规范的要求，导致各个浏览器实现自己的标准。

容器格式	视频编解码	音频编解码	**IE9+**	**Firefox5+**	**Chrome13+**
WebM	VP8	Vorbis	×	√	√
OGG	Theora	Vorbis	×	√	√
MPEG-4	H.264	AAC	√	×	疑问？

除了上面三款浏览器，还有 Safari5+支持 MPEG-4,Opera11 支持 WebM 和 OGG，通过这组数据，只要备好 MP4 和 OGG 格式的即可，但对于新的高清标准 WebM，当然是非常必要的。因为 WebM 不但清晰度高，而且免费，不受限制许可的使用源码和专利权。

目前 Chrome 浏览器，为了推广 WebM 格式的视频。声称未来将放弃 H.264 编码的视频，所以有可能在以后的版本中无法播放 MP4 的视频。当然，目前演示的版本还是支持的。

二．**video** 视频元素

以往的视频播放，需要借助 Flash 插件才可以实现。但 Flash 插件的不稳定性经常让浏览器导致崩溃，因此很多浏览器或系统厂商开始抛弃它。而取代它的正是 HTML5 的 video 元素。

属性名称	说明
src	视频资源的 URL
width	视频宽度
height	视频高度
autoplay	设置后，表示立刻开始播放视频
preload	设置后，表示预先载入视频
controls	设置后，表示显示播放控件
loop	设置后，表示反复播放视频
muted	设置后，表示视频处于静音状态
poster	指定视频数据载入时显示的图片

1.嵌入一个** WebM** 视频

```
<video src="test.webm" width="800" height="600"></video>
```

解释：<video>插入一个视频，主流的视频为.webm，.mp4，.ogg 等。src 表示资源

URL；width 表示宽度；height 表示高度。

2.附加一些属性****

```
<video src="test.webm" width="800" height="600" autoplay controls loop muted></video>
```

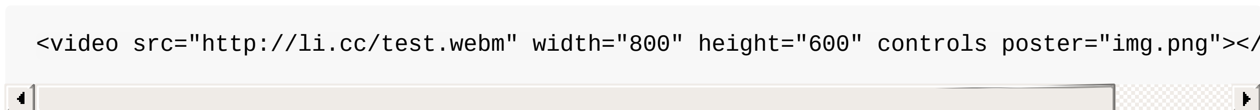
解释：autoplay 表示自动开始播放；controls 表示显示播放控件；loop 表示循环播放；muted 表示静音。

3.预加载设置****

```
<video src="http://li.cc/test.webm" width="800" height="600" controls
preload="none"></video>
```

解释：preload 属性有三个值：none 表示播放器什么都不加载；metadata 表示播放之前只能加载元数据（宽高、第一帧画面等信息）；auto 表示请求浏览器尽快下载整个视频。

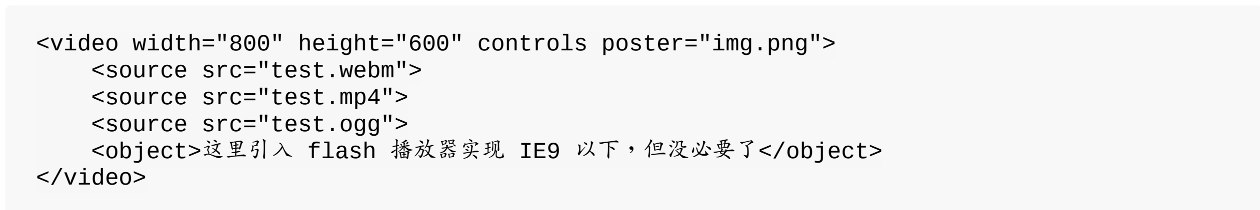
4.**使用预览图**



```
<video src="http://li.cc/test.webm" width="800" height="600" controls poster="img.png"></
```

解释：poster 属性表示在视频的第一帧，做一张预览图。

5.**兼容多个浏览器**



```
<video width="800" height="600" controls poster="img.png">
    <source src="test.webm">
    <source src="test.mp4">
    <source src="test.ogg">
    <object>这里引入 flash 播放器实现 IE9 以下，但没必要了</object>
</video>
```

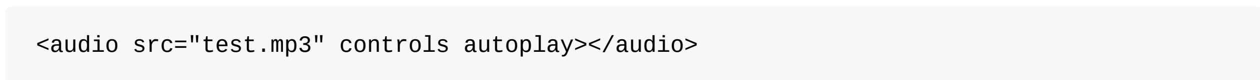
解释：通过<source>元素引入多种格式的视频，让更多的浏览器保持兼容。

二．**audio** 音频元素

和 video 元素一样，audio 元素用于嵌入音频内容，而音频元素的属性和视频元素类似。音频的支持度和视频类似，使用<source>元素引入多种格式兼容即可。主流的音频格式有：.mp3，.m4a，.ogg，.wav。

属性名称	说明
src	视频资源的 URL
autoplay	设置后，表示立刻开始播放视频
preload	设置后，表示预先载入视频
controls	设置后，表示显示播放控件

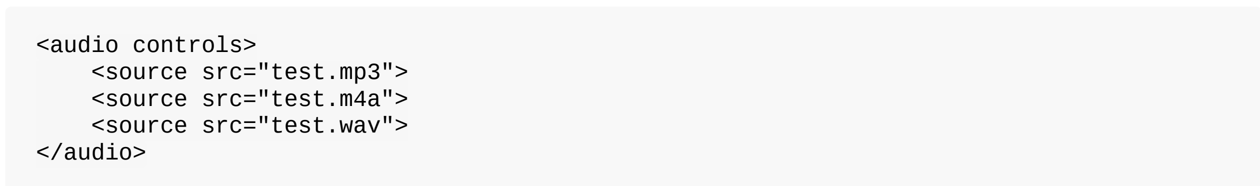
1.**嵌入一个音频**



```
<audio src="test.mp3" controls autoplay></audio>
```

解释：和嵌入视频一个道理。

2.**兼容多个浏览器**



```
<audio controls>
    <source src="test.mp3">
    <source src="test.m4a">
    <source src="test.wav">
</audio>
```

解释：略。

PS：更多设计到 API 的 JavaScript 控制，将在以后的基于 JavaScript 基础后讲解。

第 10 章 表单元素[上]

学习要点：

1.表单元素总汇

2.表单元素解析

主讲教师：李炎恢

本章主要探讨 HTML5 中表单元素，表单元素用于获取用户的输入数据。

一．表单元素总汇

在 HTML5 的表单中，提供了各种可供用户输入的表单控件。

元素名称	说明
form	表示 HTML 表单
input	表示用来收集用户输入数据的控件
textarea	表示可以输入多行文本的控件
select	表示用来提供一组固定的选项
option	表示提供提供一个选项
optgroup	表示一组相关的 option 元素
button	表示可用来提交或重置的表单按钮（或一般按钮）
datalist	定义一组提供给用户的建议值
fieldset	表示一组表单元素
legend	表示 fieldset 元素的说明性标签
label	表示表单元素的说明标签
output	表示计算结果

二．表单元素解析

1.<form>定义表单****

```
<form method="post" action="http://www.haosou.com/">

   &lt;button&gt;提交&lt;/button&gt;
&lt;/form&gt;
```

解释：<form>元素主要是定义本身是一组表单。

元素名称	说明
action	表示表单提交的页面
method	表示表单的请求方式：有 POST 和 GET 两种，默认 GET
enctype	表示浏览器对发送给服务器的数据所采用的编码格式。有三种：application/x-www-form-urlencoded（默认编码，不能将文件上传到服务器）、multipart/form-data（用于上传文件到服务器）、text/plain（未规范的编码，不建议使用，不同浏览器理解不同）
name	设置表单名称，以便程序调用
target	设置提交时的目标位置：_blank、_parent、_self、_top
autocomplete	设置浏览器记住用户输入的数据，实现自动完成表单。默认为 on 自动完成，如果不想自动完成则设置 off。
novalidate	设置是否执行客户端数据有效性检查，后面课程讲解。

//使用 get 提交数据

```
method="get"
```

//丧失自动提示功能

```
autocomplete="off"
```

//使用_blank 新建目标

```
target="_blank"
```

2.<input>表示用户输入数据****

```
<input name="user">
```

解释：<input>元素默认情况会出现一个单行文本框，有五个属性。

属性名称	说明
autofocus	让光标聚焦在某个 input 元素上，让用户直接输入
disabled	禁用 input 元素
autocomplete	单独设置 input 元素的自动完成功能
form	让表单外的元素和指定的表单挂钩提交
type	input 元素的类型，内容较多，将在下节课展开讲解
name	定义 input 元素的名称，以便接收到相应的值

//聚焦光标

```
<input name="user" autofocus>
```

//禁用 input

```
<input name="user" disabled>
```

//禁止自动完成

```
<input name="user" autocomplete="off">
```

//表单外的 input

```
<form method="get" id="register"> ... </form>
<input name="email" form="register">
```

3.<label>**添加说明标签**

```
<p><label for="user">用户名：<input id="user" name="user"></label></p>
```

解释：<label>元素可以关联 input，让用户体验更好。且更加容易设置 CSS 样式。

4.<fieldset>**对表单进行编组**

```
<fieldset>...</fieldset>
```

解释：<fieldset>元素可以将一些表单元素组织在一起，形成一个整体。

属性名称	说明
name	给分组定义一个名称
form	让表单外的分组与表单挂钩
disabled	禁用分组内的表单

5.<legend>**添加分组说明标签**

```
<fieldset>
    <legend> 注册表单 </legend>
</fieldset>
```

解释：<legend>元素给分组加上一个标题。

6.<button>**添加按钮**

```
<button type="submit"></button>
```

解释：<button>元素添加一个按钮，type 属性有如下几个值：

值名称	说明
submit	表示按钮的作用是提交表单，默认
reset	表示按钮的作用是重置表单
button	表示按钮为一般性按钮，没有任何作用

//提交表单

```
<button type="submit">提交</button>
```

//重置表单

```
<button type="reset">重置</button>
```

//普通按钮

```
<button type="button">按钮</button>
```

对于 type 属性为 submit 时，按钮还会提供额外的属性。

属性名称	说明
form	指定按钮关联的表单
formaction	覆盖 form 元素的 action 属性
formenctype	覆盖 form 元素的 enctype 属性
formmethod	覆盖 form 元素的 method 属性
formtarget	覆盖 form 元素的 target 属性
formnovalidate	覆盖 form 元素的 novalidate 属性

//表单外关联提交

```
<button type="submit" form="register">提交</button>
```

第 10 章 表单元素[中]

学习要点：

1.type 属性总汇

2.type 属性解析

主讲教师：李炎恢

本章主要探讨 HTML5 中表单中 input 元素的 type 属性，根据不同的值来显示不同的输入框。

一．**type** 属性总汇

input 元素可以用来生成一个供用户输入数据的简单文本框。在默认的情况下，什么样的数据均可以输入。而通过不同的属性值，可以限制输入的内容。

属性名称	说明
text	一个单行文本框，默认行为
password	隐藏字符的密码框
search	搜索框，在某些浏览器键入内容会出现叉标记取消
submit、reset、button	生成一个提交按钮、重置按钮、普通按钮
number、range	只能输入数值的框；只能输入在一个数值范围的框
checkbox、radio	复选框，用户勾选框；单选框，只能在几个中选一个
image、color	生成一个图片按钮，颜色代码按钮
email、tel、url	生成一个检测电子邮件、号码、网址的文本框
date、month、time、week、datetime、datetime-local	获取日期和时间
hidden	生成一个隐藏控件
file	生成一个上传控件

二．**input** 元素解析

1.type 为 text 值时**

```
<input type="text">
```

解释：当 type 值为 text 时，呈现的就是一个可以输入任意字符的文本框，这也是默认行为。并且，还提供了一些额外的属性。

属性名称	说明
list	指定为文本框提供建议值的 datalist 元素，其值为datalist 元素的 id 值
maxlength	设置文本框最大字符长度
pattern	用于输入验证的正则表达式
placeholder	输入字符的提示
readonly	文本框处于只读状态
disabled	文本框处于禁用状态
size	设置文本框宽度
value	设置文本框初始值
required	表明用户必须输入一个值，否则无法通过输入验证

//设置文本框长度

```
<input type="text" size="50">
```

//设置文本框输入字符长度

```
<input type="text" maxlength="10">
```

//设置文本框的初始值

```
<input type="text" value="初始值">
```

//设置文本框输入提示

```
<input type="text" placeholder="请输入内容">
```

//设置文本提供的建议值

```
<input list="footlist">
<datalist id="footlist">
    <option value="苹果">苹果</option>
    <option value="桔子">桔子</option>
    <option value="香蕉" label="香蕉">
    <option value="梨子">
</datalist>
```

//设置文本框内容为只读，可以提交数据

```
<input type="text" readonly>
```

//设置文本框内容不可用，不可以提交数据

```
<input type="text" disabled>
```

2.type 为** password 值时**

```
<input type="password">
```

解释：当 type 值为 password 时，一般用于密码框的输入，所有的字符都会显示星号。密码框也有一些额外属性。

属性名称	说明
maxlength	设置密码框最大字符长度
pattern	用于输入验证的正则表达式
placeholder	输入密码的提示
readonly	密码框处于只读状态
disabled	文本框处于禁用状态
size	设置密码框宽度
value	设置密码框初始值
required	表明用户必须输入同一个值

这里除了正则和验证需要放在下一节，其余和文本框一致。

3.type 为** search 时**

```
<input type="search">
```

解释：和文本框一致，在除 Firefox 浏览器的其他现代浏览器，会显示一个叉来取消搜索内容。额外属性也和 text 一致。

4.type 为** number、range 时**

```
<input type="number">
<input type="range">
```

解释：只限输入数字的文本框，不同浏览器可能显示方式不同。生成一个数值范围文本框，只是样式是拖动式。额外属性如下：

属性名称	说明
list	指定为文本框提供建议值的 datalist 元素，其值为datalist 元素的 id 值
min	设置可接受的最小值
max	设定可接受的最大值
readonly	设置文本框内容只读
required	表明用户必须输入一个值，否则无法通过输入验证
step	指定上下调节值的步长
value	指定初始值

//范围和步长

```
<input type="number" step="2" min="10" max="100">
```

5.type 为** date 系列时**

```
<input type="date">
<input type="month">
<input type="time">
<input type="week">
<input type="datetime">
<input type="datetime-local">
```

解释：实现文本框可以获取日期和时间的值，但支持的浏览器不完整。我们测试 Chrome 和 Opera 支持，其他浏览器尚未支持。所以，在获取日期和时间，目前还是推荐使用 jQuery 等前端库来实现日历功能。额外属性和 number 一致。

6.type 为** color 时**

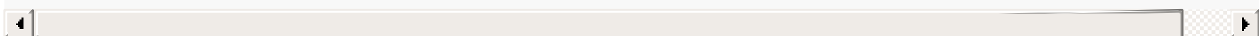
```
<input type="color">
```

解释：实现文本框获取颜色的功能，最新的现代浏览器测试后 IE 不支持，其余的都能显示一个颜色对话框提供选择。

7.type 为** checkbox、radio 时**

```
音乐<input type="checkbox"> 体育<input type="checkbox">

<input type="radio" name="sex" value="男"> 男 <input type="radio" name="sex" value="女">
```



解释：生成一个获取布尔值的复选框或固定选项的单选框。额外属性如下：

属性名称	说明
checked	设置复选框、单选框是否为勾选状态
required	表示用户必须勾选，否则无法通过验证
value	设置复选框、单选框勾选状态时提交的数据。默认为 on

//默认勾选，默认值为 1

```
<input type="checkbox" name="music" checked value="1">
```

8.type 为** submit、reset 和** **button** 时

```
<input type="submit">
```

解释：生成一个按钮，三种模式：提交、重置和一般按钮，和<button>元素相同。

值名称	说明
submit	生成一个提交按钮
reset	生成一个重置按钮
button	生成一个普通按钮

如果是 submit 时，还提供了和<button>元素一样的额外属性：formaction、formenctype、formmethod、formtarget 和 formnovalidate。

9.type 为** image 时**

```
<input type="image" src="img.png">
```

解释：生成一个图片按钮，点击图片就实现提交功能，并且传送了分区响应数据。图片按钮也提供了一些额外属性。

属性名称	说明
src	指定要显示图像的 URL
alt	提供图片的文字说明
width	图像的长度
height	图像的高度
提交额外属性	formaction、formenctype、formmethod、formtarget和 formnovalidate。

10.type 为** email、tel、url 时**

```
<input type="email">
<input type="tel">
<input type="url">
```

解释：email 为电子邮件格式、tel 为电话格式、url 为网址格式。额外属性和 text 一致。但对于这几种类型，浏览器支持是不同的。email 支持比较好，现在浏览器都支持格式验证；tel 基本不支持；url 支持一般，部分浏览器只要检测到 http://就能通过。

11.type 为** hidden 时**

```
<input type="hidden">
```

解释：生成一个隐藏控件，一般用于表单提交时关联主键 ID 提交，而这个数据作为隐藏存在。

12.type 为** file 时**

```
<input type="file">
```

解释：生成一个文件上传控件，用于文件的上传。额外提供了一些属性：

属性名称	说明
accept	指定接受的 MIME 类型
required	表明用户必须提供一个值，否则无法通过验证

```
accept="image/gif, image/jpeg, image/png"
```

第 10 章 表单元素[下]

学习要点：

1.其他元素

2.输入验证

主讲教师：李炎恢

本章主要探讨 HTML5 中表单中剩余的其他元素，然后重点了解一下表单的输入验证功能。

一．其他元素

表单元素还剩下几个元素没有讲解，包括下拉框列表 select、多行文本框 textarea、和 output 计算结果元素。

元素名称	说明
select	生成一个下拉列表进行选择
optgroup	对 select 元素进行编组
option	select 元素中的项目
textarea	生成一个多行文本框
output	表示计算结果

1.**生成下拉列表**

```
<select name="fruit">
    <option value="1">苹果</option>

    &lt;option value="2"&gt;橘子&lt;/option&gt;

    &lt;option value="3"&gt;香蕉&lt;/option&gt;
&lt;/select&gt;
```

解释：<select>下拉列表元素至少包含一个<option>子元素，才能形成有效的选项列表。<select>元素包含两个子元素<option>项目元素和<optgroup>分组元素，还包含了一些额外属性。

属性名称	说明
name	设定提交时的名称
disabled	将下拉列表禁用
form	将表单外的下拉列表与某个表单挂钩
size	设置下拉列表的高度
multiple	设置是否可以多选
autofocus	获取焦点
required	选择验证，设置后必须选择才能通过

//设置高度并实现多选

```
<select name="fruit" size="30" multiple>
```

//默认首选

```
<option value="2" selected>橘子</option>
```

//使用 optgroup 进行分组，label 为分组名称，disabled 可以禁用分组

```
<optgroup label="水果类">
    <option value="1">苹果</option>
    <option value="2" selected>橘子</option>
    <option value="3" label="香蕉">香蕉</option>
</optgroup>
```

2.多行文本框****

```
<textarea name="content">请留下您的建议！ </textarea>
```

解释：生成一个可变更大小的多行文本框。属性如下：

属性名称	说明
name	设定提交时的名称
form	将表单外的多行文本框与某个表单挂钩
readonly	设置多行文本框只读
disabled	将多行文本框禁用
maxlength	设置最大可输入的字符长度
autofocus	获取焦点
placeholder	设置输入时的提示信息
rows	设置行数
cols	设置列数
wrap	设置是否插入换行符，有 soft 和 hard 两种
required	设置必须输入值，否则无法通过验证

//设置行高和列宽，设置插入换行符

```
<textarea name="content" rows="20" cols="30" wrap="hard"></textarea>
```

3.计算结果

```
<form oninput="res.value = num1.valueAsNumber * num2.valueAsNumber">
    <input type="number" id="num1"> x <input type="number" id="num2">
    <output for="num1 num2" name="res">
</form>
```

解释：output 就是计算两个文本框之间的值，其实就是内嵌了 JavaScript 功能。

二．输入验证

HTML5 对表单提供了输入验证检查方式，但这种验证还是比较简陋的，并且不同的浏览器支持的成熟度还不同。在大部分情况下，可能还是要借助 jQuery 等前端库来实现丰富的验证功能和显示效果。

//必须填写一个值

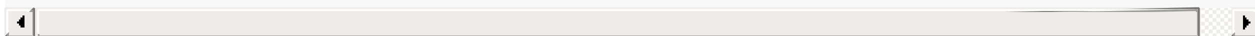
```
<input type="text" required>
```

//限定在某一个范围内

```
<input type="number" min="10" max="100">
```

//使用正则表达式

```
<input type="text" placeholder="请输入区号+座机" required pattern="^[\d]{2,4}\-[\d]{6,8}$">
```



//禁止表单验证

```
<form action="http://li.cc" novalidate>
```

第 11 章 全局属性和其他

学习要点：

1.实体

2.元数据

3.全局属性

主讲教师：李炎恢

本章主要探讨 HTML5 中的 HTML 实体、以及 HTML 核心构成的元数据，最后了解一下 HTML 中的全局属性。

一．实体

HTML 实体就是将有特殊意义的字符通过实体代码显示出来。

显示结果	实体名称	实体编号	描述
			空格
<	<	<	小于号
>	>	>	大于号
&	&	&	和号
"	"	"	引号
'	'	'	撇号
¢	¢	¢	分
£	£	£	镑
¥	¥	¥	日圆
€	€	€	欧元
§	§	§	小节
©	©	©	版权
®	®	®	注册商标
™	™	™	商标
×	×	×	乘号
÷	÷	÷	除号

二．元数据

<meta>元素可以定义文档中的各种元数据，而且一个 HTML 页面可以包含多个<meta>元素。

1.**指定名/值元数据对**

```
<meta name="author" content="bnbbs">
<meta name="description" content="这是一个 HTML5 页面">

&lt;meta name="keywords" content="html5,css3,响应式"&gt;

&lt;meta name="generator" content="sublime text 3"&gt;
```

元数据名称	说明
author	当前页面的作者
description	当前页面的描述
keywords	当前页面的关键字
generator	当前页面的编码工具

2.声明字符编码

```
<meta charset="utf-8">
```

3.模拟 **HTTP** 标头字段

//5 秒跳转到指定 URL

```
<meta http-equiv="refresh" content="5;http://li.cc">
```

//另一种声明字符编码

```
<meta http-equiv="content-type" content="text/html charset=utf-8">
```

属性值	说明
refresh	重新载入当前页面，或指定一个 URL。单位秒。
content-type	另一种声明字符编码的方式

三．全局属性

在此之前，我们涉及到的元素都讲解了它的局部数据，当然也知道一些全局属性，比如id。全局属性是所有元素共有的行为，HTML5 还提供了一些其他的全局属性。

1.id 属性

```
<p id="abc">段落</p>
```

解释：id 属性给元素分配一个唯一标识符。这种标识符通常用来给 CSS 和 JavaScript 调用选择元素。一个页面只能出现一次同一个 id 名称。

2.class 属性

```
<p class="abc">段落</p>
&lt;p class="abc"&gt;段落&lt;/p&gt;
&lt;p class="abc"&gt;段落&lt;/p&gt;
```

解释：class 属性用来给元素归类。通过是文档中某一个或多个元素同时设置 CSS 样式。

3.accesskey 属性

```
<input type="text" name="user" accesskey="n">
```

解释：快捷键操作，windows 下 alt+指定键，前提是浏览器 alt 并不冲突。

4.contenteditable 属性

```
<p contenteditable="true">我可以修改吗</p>
```

解释：让文本处于可编辑状态，设置 true 则可以编辑，false 则不可编辑。或者直接设置属性。

5.dir 属性

```
<p dir="rtl">文字到右边了</p>
```

解释：让文本从左到右（ltr），还是从右到左（rtl）。

6.hidden 属性

```
<p hidden>文字到右边了</p>
```

解释：隐藏元素。

7.lang 属性

```
<p lang="en">HTML5</p>
```

解释：可以局部设置语言。

8.title 属性

```
<p title="HTML5 教程">HTML5</p>
```

解释：对元素的内容进行额外的提示。

9.tabindex 属性

```
<input type="text" name="user" tabindex="2">

&lt;input type="text" name="user" tabindex="1"&gt;
```

解释：表单中按下 tab 键，实现获取焦点的顺序。如果是-1，则不会被选中。

10.style 属性

```
<p style="color:red;">CSS 样式</p>
```

解释：设置元素行内 CSS 样式。

第 12 章 CSS 入门

学习要点：

1.使用 CSS

2.三种方式

3.层叠和继承

主讲教师：李炎恢

本章主要探讨 HTML5 中 CSS（层叠样式表），它是用来对 HTML 文档外观的表现形式进行排版和格式化。

一．使用 **CSS**

CSS 样式由一条或多条以分号隔开的样式声明组成。每条声明的样式包含着一个 CSS 属性和属性值。

```
<p style="color:red;font-size:50px;">这是一段文本</p>
```

解释：style 是行内样式属性。color 是颜色属性，red 是颜色属性值；font-size是字体大小属性，50px 是字体大小属性值。

二．三种方式

创建 CSS 样式表有三种方式：1.元素内嵌样式；2.文档内嵌样式；3.外部引入样式。

1.**元素内嵌样式**

```
<p style="color:red;font-size:50px;">这是一段文本</p>
```

解释：即在当前元素使用 style 属性的声明方式。

2.**文档内嵌样式**

```
<style type="text/css"> p {     color: blue; font-size: 40px;
}
&lt;/style&gt;

&lt;p&gt;这是一段文本&lt;/p&gt;
```

解释：在<head>元素之间创建<style>元素，通过选择器的方式调用指定的元素并设置相关 CSS。

3.**外部引用样式**

```
<link rel="stylesheet" type="text/css" href="style.css">
```

//style.css

```
@charset "utf-8";
p {    color: green;    font-size: 30px;
}
```

解释：很多时候，大量的 HTML 页面使用了同一个组 CSS。那么就可以将这些 CSS 样式保存在一个单独的.css 文件中，然后通过<link>元素去引入它即可。@charset "utf-8" 表明设置 CSS 的字符编码，如果不写默认就是 utf-8。如果有多个.css 文件，可以使用 @import 导入方式引入.css 文件。只不过，性能不如多个<link>链接。

三．层叠和继承

所谓的样式表层叠：指的是同一个元素通过不同方式设置样式表产生的样式重叠。样式表继承：指的是某一个被嵌套的元素得到它父元素样式。还有一种样式叫浏览器样式，是这个元素在这个浏览器运行时默认附加的样式。

1.**浏览器样式**

```
<b>这个元素隐含加粗样式</b>
<span style="font-weight:bold;">这个元素通过 style 加粗</span>
```

解释：<b>元素就是具有加粗的隐含样式，而<span>元素没有任何隐含样式，通过 style 属性设置样式。

2.**样式表层叠**

样式表层叠通过五种方式进行，如果样式相同，那么比如会产生冲突替换。这时，它的优先级顺序就显的比较重要。以下优先级从低到高：

(1).浏览器样式（元素自身携带的样式）；

(2).外部引入样式（使用<link>引入的样式）；

(3).文档内嵌样式（使用<style>元素设置）；

(4).元素内嵌样式（使用 style 属性设置）。

//元素内嵌

```
<p style="color:red;font-size:30px;">我将被三种方式叠加样式</p>
```

//文档内嵌

```
<style type="text/css">

p {     color:blue; font-weight: bold;
}
&lt;/style&gt;
```

//外部引入

```
@charset "utf-8";
p {     color: green;     font-style: italic;
}
```

如果某一个样式被优先级高的给替换掉了，却又想执行这个样式方案，可以将这个方案标记成重要样式（important）。

//强行设置最高优先级

```
color: green !important;
```

3.**样式继承**

如果某一个元素并没有设置父元素相关的样式，那么就会使用继承机制将父元素的样式集成下来。

//<b>元素继承了<p>元素的样式

```
<p style="color:red;">这是<b>HTML5</b></p>
```

样式继承只适用于元素的外观（文字、颜色、字体等），而元素在页面上的布局样式则不会被继承。如果继承这种样式，就必须使用强制继承：inherit。

//强制继承布局样式

```
<p>这是<b>HTML5</b></p>

<style type="text/css">

p {     border: 1px solid red;
} b {     border : inherit;
}
&lt;/style&gt;
```

第 13 章 CSS 选择器[上]

学习要点：

1.选择器总汇

2.基本选择器

3.复合选择器

4.伪元素选择器

主讲教师：李炎恢

本章主要探讨 HTML5 中 CSS 选择器，通过选择器定位到想要设置样式的元素。目前 CSS 选择器的版本已经升级至第三代，即 CSS3 选择器。CSS3 选择器提供了更多、更丰富的选择器方式，主要分为三大类。

一．选择器总汇

本节课主要涉及到三种选择器：基本选择器、复合选择器和伪元素选择器，具体如下：

选择器	名称	说明	CSS 版本
*	通用选择器	选择所有元素	2
<type>	元素选择器	选择指定类型的元素	1
#<id>	id 选择器	选择指定 id 属性的元素	1
.<class>	class 选择器	选择指定 class 属性的元素	1
[attr]系列	属性选择器	选择指定 attr 属性的元素	2 ~ 3
s1,s2,s3...	分组选择器	选择多个选择器的元素	1
s1 s2	后代选择器	选择指定选择器的后代元素	1
s1 > s2	子选择器	选择指定选择器的子元素	2
s1 + s2	相邻兄弟选择器	选择指定选择器相邻的元素	2
s1 ~ s2	普通兄弟选择器	选择指定选择器后面的元素	3
::first-line	伪元素选择器	选择块级元素文本的首行	1
::first-letter	伪元素选择器	选择块级元素文本的首字母	1
::before	伪元素选择器	选择元素之前插入内容	2
::after	伪元素选择器	选择元素之后插入内容	2

二．基本选择器

使用简单且频率高的一些选择器归类为基本选择器。

1.通用选择器****

```
* { border: 1px solid red;
}
```

解释："*"号选择器是通用选择器，功能是匹配所有 html 元素的选择器包括<html>和<body>标签。可以使用如下元素标记测试效果：

```
<p>段落</p>

&lt;b&gt;加粗&lt;/b&gt;

&lt;span&gt;无&lt;/span&gt;
```

通用选择器会将所有元素匹配并配置样式，这是把双刃剑，好处就是非常方便，坏处就是将不必要的元素也配置了。目前为止，还不存在所有元素都必须配置的样式，所以，一般来说，不常用。

2.元素选择器****

```
p { color: red;
}
```

```
<p>段落</p>
```

解释：直接使用元素名称作为选择器名即可。

3.ID 选择器

```
#abc { font-size: 20px;
}
```

```
<p id="abc">段落</p>
```

解释：通过对元素设置全局属性 id，然后使用#id 值来选择页面唯一元素。

4.类选择器****

```
.abc { border: 1px solid red;
}
```

```
<b class="abc">加粗</b>

&lt;span class="abc"&gt;无&lt;/span&gt;
```

解释：通过对元素设置全局属性 class，然后使用.class 值选择页面一个或多个元素。

```
b.abc { border: 1px solid red;
}
```

解释：也可以使用“元素.class 值”的形式，限定某种类型的元素。

```
<span class="abc edf">无</span>
```

解释：类选择器还可以调用多个样式，中间用空格隔开进行层叠。

5.**属性选择器**

//所需 CSS2 版本

```
[href] { color: orange;
}
```

解释：属性选择器，直接通过两个中括号里面包含属性名即可。当然，还有更多扩展的属性选择器。

//所需 CSS2 版本

```
[type="password"] { border: 1px solid red;
}
```

解释：匹配属性值的属性选择器。

//所需版本 CSS3

```
[href^="http"] { color: orange;
}
```

解释：属性值开头匹配的属性选择器。

//所需版本 CSS3

```
[href$=".com"] { color: orange;
}
```

解释：属性值结尾匹配的属性选择器。

//所需版本 CSS3

```
[href*="baidu"] { color: orange;
}
```

解释：属性值包含指定字符的属性选择器。

//所需版本 CSS2

```
[class~="edf"] { font-size: 50px;
}
```

解释：属性值具有多个值时，匹配其中一个值的属性选择器。

//所需版本 CSS2

```
[lang|="en"] { color: red;
}
```

解释：属性值具有多个值且使用"-"号连接符分割的其中一个值的属性选择器。比如

```
<i lang="en-us">HTML5</i>
```

二．复合选择器

将不同的选择器进行组合形成新的特定匹配，我们称为复合选择器。

1.分组选择器****

```
p,b,i,span { color: red;
}
```

解释：将多个选择器通过逗号分割，同时设置一组样式。当然，不但可以分组元素选择器，还可以使用 ID 选择器、类选择器、属性选择器混合使用。

2.后代选择器****

```
p b { color: red;
}
```

解释：选择<p>元素内部所有<b>元素。不在乎<b>的层次深度。当然，后代选择器也可以混合使用 ID 选择器、类选择器、属性选择器。

3.子选择器****

```
ul > li { border: 1px solid red;
}
```

```
<ul>
    <li> 我是儿子 <ol>
            <li> 我是孙子 </li>
            <li> 我是孙子 </li>
        </ol>
    </li>
    <li> 我是儿子 </li>
    <li> 我是儿子 </li>
</ul>
```

解释：子选择器类似与后代选择器，而最大的区别就是子选择器只能选择父元素向下一级的元素，不可以再往下选择。

4.**相邻兄弟选择器**

```
p + b { color: red;
}
```

解释：相邻兄弟选择器匹配和第一个元素相邻的第二个元素。

5.**普通兄弟选择器**

```
p ~ b { color: red;
}
```

解释：普通兄弟选择器匹配和第一个元素后面的所有元素。

三·伪元素选择器

伪选择器分为两种第一种是下节伪类选择器，还有一种就是伪元素选择器。这两种选择器特性上比较容易混淆，在 CSS3 中为了区分，伪元素前置两个冒号（::），伪类前置一个冒号（:）。

1.::first-line 块级首行

```
::first-line { color: red;
}
```

解释：块级元素比如<p>、<div>等的首行文本被选定。如果想限定某种元素，可以加上前置 p::first-line。

2.::first-letter 块级首字母

```
::first-letter { color: red;
}
```

解释：块级元素的首行字母。

3.::before 文本前插入

```
a::before { content: '点击-';
}
```

解释：在文本前插入内容。

4.::after 文本后插入

```
a::before { content: '-请进';
}
```

解释：在文本后插入内容。

第 14 章 CSS 颜色与度量单位

学习要点：

1.颜色表方案

2.度量单位

主讲教师：李炎恢

本章主要探讨 HTML5 中 CSS 颜色和度量单位等问题，包括颜色的选取方式、相对长度和绝对长度等。

一．颜色表方案

颜色的表现形式主要有三种方式：颜色名称、十六进制代码和十进制代码。

```
p { color: red;
}
```

解释：这是将一个段落内的文字设置为红色，采用的是英文颜色名称。问题是，其他各种颜色我们将如何设置？

在古老的 HTML4 时，颜色名称只有 16 种。

颜色名称	十六进制代码	十进制代码	含义
black	#000000	0,0,0	黑色
silver	#c0c0c0	192,192,192	银灰色
gray	#808080	128,128,128	灰色
white	#ffffff	255,255,255	白色
maroon	#800000	128,0,0	栗色
red	#ff0000	255,0,0	红色
purple	#800080	128,0,128	紫色
fuchsia	#ff00ff	255,0,255	紫红
green	#008000	0,128,0	绿色
lime	#00ff00	0,255,0	闪光绿
olive	#808000	128,128,0	橄榄色
yellow	#ffff00	255,255,0	黄色
navy	#000080	0,0,128	海军蓝
blue	#0000ff	0,0,255	蓝色
teal	#008080	0,128,128	水鸭色
aqua	#00ffff	0,255,255	浅绿色

当然，目前颜色名称远远不止这些，可以搜索更多的 HTML 颜色表或 CSS 颜色表查阅。这里提供一些页面如下：

http://xh.5156edu.com/page/z1015m9220j18754.html

http://finle.me/colors.html

http://www.w3school.com.cn/tags/html_ref_colornames.asp

在上面的表格中，我们也罗列出对应的十六进制和十进制颜色表示方法。使用方法如下：

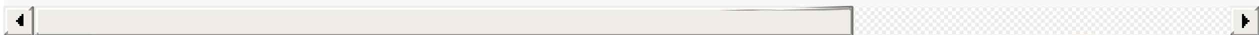
//红色的十六进制方案

```
p { color: #ff0000;
}
```

十进制表示方法就比较多样化，有四种方案：

函数	说明	示例
rgb(r,g,b)	用 RGB 模型表示颜色	rgb(0,128,128)
rgba(r,g,b,a)	同上，a 表示透明度 0~1 之间	rgba(0,128,128,0.5)
hsl(h,s,l)	用 HSL 模型（色相、饱和度和透明度）来表示颜色	hsl(120,100%,30%)
hsla(h,s,l,a)	同上，a 表示透明度 0~1 之间	hsla(120,100%,30%,0.5)

```
p { color: rgb(112, 128, 114); color: rgba(0, 128, 128, 0.5); color: hsl(120, 100%, 30%);
}
```



目前又有一个疑问，这些值从哪里获取。除了颜色表之外，想要微调自己的颜色值。我们可以使用 photoshop 等平面设计软件的调色板获取相应的值。



二．度量单位

在 CSS 长度设置中，我们经常需要使用到度量单位，即以什么样的单位设计我们的字体或边框长度。而在 CSS 中长度单位又分为绝对长度和相对长度。

绝对长度指的是现实世界的度量单位，CSS 支持五种绝对长度单位。	
绝对长度单位	
单位标识符	说明
in	英寸
cm	厘米
mm	毫米
pt	磅
pc	pica
相对长度指的是依托其他类型的单位，也是五种。	
相对长度单位	
单位标识符	说明
em	与元素字号挂钩
ex	与元素字体的“x 高度”挂钩
rem	与根元素的字号挂钩
px	像素，与分辨率挂钩
%	相对另一值的百分比

下面我们使用一些常用的单位作为演示，而不做演示的基本用不到了。

//em 相对单位

```
p { margin: 0; padding: 0; background: silver; font-size: 15px; height: 2em;
}
```

解释：em 是相对单位，与字号大小挂钩，会根据字体大小改变自己的大小，灵活性很高。

//px 相对单位，绝对特性

```
p { margin: 0; padding: 0; background: silver; font-size: 15px; height: 55px;
}
```

解释：虽然 px 也是相对单位，但由于和分辨率挂钩，导致他其实就变成一个绝对单位了，自然灵活性没有 em 高，但是使用难度较低，且大量的开发者习惯性使用它。

//%百分比

```
p { margin: 0; padding: 0; background: silver; font-size: 200%; width: 50%;
}
```

解释：长度比较好理解，就是挂钩它所在区块的宽度。而 font-size 则是继承到的原始大小的百分比。

第 15 章 CSS 文本样式[上]

学习要点：

1.字体总汇

2.字体设置

3.Web 字体

主讲教师：李炎恢

本章主要探讨 HTML5 中 CSS 文本样式，通过文本样式的设置，更改字体的大小、样式以及文本的方位。

一．字体总汇

本节课，我们重点了解一下 CSS 文本样式中字体的一些设置方法，样式表如下：

属性名	说明	CSS 版本
font-size	设置字体的大小	1
font-variant	设置英文字体是否转换为小型大写	1
font-style	设置字体是否倾斜	1
font-weight	设置字体是否加粗	1
font-family	设置 font 字体	1
font	设置字体样式复合写法	1 ~ 2
@font-face	设置 Web 字体	3

二．字体设置

我们可以通过 CSS 文本样式来修改字体的大小、样式以及形态。

1.font-size

```
p { font-size: 50px;
}
```

解释：设置文本的大小。属性值如下表：

值	说明
xx-small	设置字体大小。每个值从小到大的固定值。
x-small	
small	
medium	
large	
x-large	
xx-large	
smaller	设置字体相对于父元素字体的大小
larger	
数字+px	使用 CSS 像素长度设置字体大小
数字+%	使用相对于父元素字体的百分比大小

//先设置父元素字体大小

```
body { font-size: 50px;
}
```

//再设置相对小一些

```
p { font-size: smaller;
}
```

2.font-variant

```
p { font-variant: small-caps;
}
```

解释：设置字体是否以小型大写字母显示。

值	说明
normal	表示如果以小型大写状态，让它恢复小写状态。
small-caps	让小写字母以小型大写字母显示。

//先让父元素设置小型大写

```
body { font-variant: small-caps;
}
```

//让子元素设置恢复小写

```
p { font-size: 50px; font-variant: normal;
}
```

3.font-style

```
p { font-style: italic;
}
```

解释：设置字体是否倾斜。

值	说明
normal	表示让倾斜状态恢复到正常状态。
italic	表示使用斜体。
oblique	表示让文字倾斜。区别在没有斜体时使用。

```
p { font-weight: bold;
}
```

解释：设置字体是否加粗。

值	说明
normal	表示让加粗的字体恢复正常。
bold	粗体
bolder	更粗的字体
lighter	轻细的字体
100 ~ 900 之间的数字	600 及之后是加粗，之前不加粗

在目前计算机和浏览器显示中，只有 bold 加粗，其他更粗更细，目前体现不出来。

5.font-family

```
p { font-family: 微软雅黑;
}
```

解释：使用指定字体名称。这里使用的字体是浏览者系统的字体。有时为了兼容很多浏览者系统的字体，可以做几个后备字体。

//备用字体

```
p { font-family: 楷体,微软雅黑,宋体;
}
```

6.font

```
p { font: 50px 楷体;
}
```

解释：字体设置简写组合方式。格式如下：[是否倾斜|是否加粗|是否转小型大写] 字体大小 字体名称；

三．**Web** 字体

虽说可以通过备用字体来解决用户端字体缺失问题，但终究用户体验不好，且不一定备用字体所有用户都安装了。所以，现在 CSS 提供了 Web 字体，也就是服务器端字体。

//服务器提供字体

```
@font-face { font-family: abc; src: url('BrushScriptStd.otf');
} p { font-size: 50px; font-family: abc;
}
```

英文字体文件比较小，而中文则很大。所以，中文如果想用特殊字体可以使用图片。大面积使用特殊中文字体，就不太建议了。

第 15 章 CSS 文本样式[下]

学习要点：

1.文本总汇

2.文本样式

3.文本控制

主讲教师：李炎恢

本章主要探讨 HTML5 中 CSS 文本样式，通过文本样式的设置，更改字体的大小、样式以及文本的方位。

一．文本总汇

本节课，我们重点了解一下 CSS 文本样式中文本内容的一些设置方法，样式表如下：

属性名	说明	CSS 版本
text-decoration	装饰文本出现各种划线。	1
text-transform	将英文文本转换大小写。	1
text-shadow	给文本添加阴影	3
text-align	设置文本对齐方式	1,3
white-space	排版中的空白处理方式	1
letter-spacing	设置字母之间的间距	1
word-spacing	设置单词之间的间距	1
line-height	设置行高	1
word-wrap	控制段词	3
text-indent	设置文本首行的缩进	1

二．文本样式

CSS 文本样式有三种：文本装饰、英文大小写转换和文本阴影。

1.text-decoration

```
p { text-decoration: underline;
}
```

解释：设置文本出现下划线。属性值如下表：

值	说明
none	让本身有划线装饰的文本取消掉
underline	让文本的底部出现一条下划线
overline	让文本的头部出现一条上划线
line-through	让文本的中部出现一条删除划线
blink	让文本进行闪烁，基本不支持了

//让本来有下划线的超链接取消

```
a { text-decoration: none;
}
```

2.text-transform

```
p { text-transform: uppercase;
}
```

解释：设置英文文本转换为大小写。

值	说明
none	将已被转换大小写的值恢复到默认状态
capitalize	将英文单词首字母大写
uppercase	将英文转换为大写字母
lowercase	将英文转换为小写字母

3.text-shadow

```
p { text-shadow: 5px 5px 3px black;
}
```

解释：给文本添加阴影。其中四个值，第一个值：水平偏移；第二个值：垂直偏移；第三个值：阴影模糊度（可选）；第四个值：阴影颜色（可选）。

三．文本控制

CSS 文本样式中还有一组对文本进行访问、形态进行控制的样式。

1.text-align

```
p { text-align: center;
}
```

解释：指定文本的对齐方式。

值	说明
left	靠左对齐，默认
right	靠右对齐
center	居中对齐
justify	内容两端对齐
start	让文本处于结束的边界
end	让文本处于结束的边界

start 和 end 属于 CSS3 新增的功能，但目前 IE 和 Opera 尚未支持。

2.white-space

```
p { white-space: nowrap;
}
```

解释：处理空白排版方式。

值	说明
normal	默认值，空白符被压缩，文本自动换行
nowrap	空白符被压缩，文本不换行
pre	空白符被保留，遇到换行符则换行
pre-line	空白符被压缩，文本会在排满或遇换行符换行
pre-wrap	空白符被保留，文本会在排满或遇换行符换行

3.letter-spacing

```
p { letter-spacing: 4px;
}
```

解释：设置文本之间的间距。

值	说明
normal	设置默认间距
长度值	比如：“数字”+“px”

4.word-spacing

```
p { word-spacing: 14px;
}
```

解释：设置英文单子之间的间距。

值	说明
normal	设置默认间距
长度值	比如："数字"+"px"

5.line-height

```
p { line-height: 200%;
}
```

解释：设置段落行高。

值	说明
normal	设置默认间距
长度值	比如："数字"+"px"
数值	比如：1,2,3
%	比如：200%

6.word-wrap

```
p { word-wrap: break-word;
}
```

解释：让过长的英文单词断开。

值	说明
normal	单词不断开
break-word	断开单词

7.text-indent

```
p { text-indent: 20px;
}
```

解释：设置文本首行的缩进。

值	说明
normal	设置默认间距
长度值	比如："数字"+"px"

第 16 章 CSS 盒模型[上]

学习要点：

1.元素尺寸

2.元素内边距

3.元素外边距

4.处理溢出

主讲教师：李炎恢

本章主要探讨 HTML5 中 CSS 盒模型，学习怎样了解元素的外观配置以及文档的整体布局。

一．元素尺寸

CSS 盒模型中最基础的就是设置一个元素的尺寸大小。有三组样式来配置一个元素的尺寸大小，样式表如下：

属性	值	说明	CSS 版本
width	auto、长度值或百分比	设置元素的宽度	1
height	auto、长度值或百分比	设置元素的高度	1
min-width	auto、长度值或百分比	设置元素最小宽度	2
min-height	auto、长度值或百分比	设置元素最小高度	2
max-width	auto、长度值或百分比	设置元素最大宽度	2
max-height	auto、长度值或百分比	设置元素最大高度	2

//设置元素尺寸

```
div { width: 200px; height: 200px;
}
```

解释：设置元素的固定尺寸。

//限制元素尺寸

```
div { min-width: 100px; min-height: 100px; max-width: 300px; max-height: 300px;
}
```

解释：这一组主要是应对可能动态产生元素尺寸变大变小的问题，从而限制它最大和最小的值。

//auto 自适应

```
div { width: auto; height: auto;
}
```

解释：auto 是默认值，width 在 auto 下是 100%的值；height 在 auto 下是自适应。

//百分比方式

```
#a { background: silver; width: 200px; height: 200px;
} #b { background: gray; width: 80%; height: 80%;
}
```

```
<div id="a">
    <div id="b"> 我是 html5 </div>
</div>
```

解释：百分比就是相对于父元素长度来衡定的。

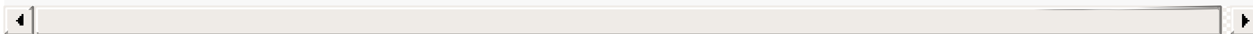
二．元素内边距

CSS 盒模型中可以设置元素内部边缘填充空白的大小，我们成为内边距。样式表如下：

属性	值	说明	CSS 版本
padding-top	长度值或百分比	设置顶部内边距	1
padding-bottom	长度值或百分比	设置底部内边距	1
padding-left	长度值或百分比	设置左边内边距	1
padding-right	长度值或百分比	设置右边内边距	1
padding	1～4 个长度值或百分比	简写属性	1

//设置四个内边距

```
div { padding-top: 10px; padding-bottom: 10px; padding-left: 10px; padding-right: 10px;
}
```



//简写形式，分别为上 10px、右 10px、下 10px、左 10px

```
div { padding: 10px 10px 10px 10px;
}
```

//简写形式，分别为上 10px，左右 50px，下 200px

```
div { padding: 10px 50px 200px;
}
```

//简写形式，分别是上下 10px，左右 20px

```
div { padding: 10px 20px;
}
```

//简写形式：上下左右均 10px

```
div { padding: 10px;
}
```

三．元素外边距

CSS 盒模型中可以设置元素外部边缘填充空白的大小，我们成为外边距。样式表如下：

属性	值	说明	CSS 版本
margin-top	长度值或百分比	设置顶部内边距	1
margin-bottom	长度值或百分比	设置底部内边距	1
margin-left	长度值或百分比	设置左边内边距	1
margin-right	长度值或百分比	设置右边内边距	1
margin	1 ~ 4 长度值或百分比	简写属性	1

//设置四个内边距

```
div { margin-top: 10px; margin-bottom: 10px; margin-left: 10px; margin-right: 10px;
}
```

//简写形式，分别为上 10px、右 10px、下 10px、左 10px

```
div { margin: 10px 10px 10px 10px;
}
```

//简写形式，分别为上 10px，左右 50px，下 200px

```
div { margin: 10px 50px 200px;
}
```

//简写形式，分别是上下 10px，左边 20px

```
div { margin: 10px 20px;
}
```

//简写形式：上下左右均 10px

```
div { margin: 10px;
}
```

四．处理溢出

当设置了元素固定尺寸且内容过大时，就会出现溢出的问题。溢出主要朝两个方向：右侧和底部。我们可以通过 overflow 系列样式来控制它。

属性	值	说明	CSS 版本
overflow-x	溢出值	设置水平方向的溢出	3
overflow-y	溢出值	设置垂直方向的溢出	3
overflow	溢出值	简写属性	2

溢出处理主要有四种处理值：

值	说明
auto	浏览器自行处理溢出内容。如果有溢出内容，就显示滚动条，否则就不显示滚动条。
hidden	如果有溢出的内容，直接剪掉。
scroll	不管是否溢出，都会出现滚动条。但不同平台和浏览器显示方式不同。
visible	默认值，不管是否溢出，都显示内容。

//设置溢出为 auto 值

```
div { overflow-x: auto;
}
```

第 16 章 CSS 盒模型[下]

学习要点：

1.元素可见性

2.元素盒类型

3.元素的浮动

主讲教师：李炎恢

本章主要探讨 HTML5 中 CSS 盒模型，学习怎样了解元素的外观配置以及文档的整体布局。

一．元素可见性

使用 visibility 属性可以实现元素的可见性，这种样式一般可以配合 JavaScript 来实现效果。样式表如下：

属性	值	说明	CSS 版本
visible	默认值，元素在页面上可见。	2	
visibility	hidden	元素不可见，但会占据空间。	2
collapse	元素不可见，隐藏表格的行与列，如果不是表格，则和 hidden 一样。	2	

//设置元素隐藏，但占位

```
div { visibility: hidden;
}
```

//隐藏表格的行或列，但不占位，Chrome 和 Opera 不支持

```
table tr:first-child { visibility: collapse;
}
```

二．元素盒类型

CSS 盒模型中的 display 属性，可以更改元素本身盒类型。那么元素有哪些盒类型呢？主要有：1.块级元素（区块）；2 行内元素（内联）；3 行内-块级元素（内联块）；4.隐藏元素。

1.**块级元素**

所谓块级元素，就是能够设置元素尺寸、隔离其他元素功能的元素。比如：<div>、<p>等文档元素。

2.**行内元素**

所谓行内元素，不能够设置元素尺寸，它只能自适应内容、无法隔离其他元素，其它元素会紧跟其后。比如：<span>、<b>等文本元素。

3.**行内-块元素**

所谓行内-块元素，可以设置元素尺寸，但无法隔离其他元素的元素。比如<img>。

属性	值	说明	CSS 版本
block	盒子为块级元素	1	
display	inline	盒子为行内元素	1
inline-block	盒子为行内-块元素	2	
none	盒子不可见，不占位	1	

//将行内元素转成块级元素

```
span { background: silver; width: 200px; height: 200px; display: block;
}
```

//将块级元素转换成行内元素

```
div { background: silver; width: 200px; height: 200px; display: inline;
}
```

//将块级元素转化成行内-块元素

```
div { background: silver; width: 200px; height: 200px; display: inline-block;
}
```

//将元素隐藏且不占位

```
div { display: none;
}
```

display 属性还有非常多的值，有些后面部分讲解，而有些支持度不好或者尚不支持，从而省略。有兴趣的，可以参考 CSS3 手册。

三．元素的浮动

CSS 盒模型有一种叫浮动盒，就是通过 float 属性建立盒子的浮动方向，样式表如下：

属性	值	说明	CSS 版本
	left	浮动元素靠左	1
float	right	浮动元素靠右	1
	none	禁用浮动	1

//实现联排效果

```
#a { background: gray; float: left;
} #b { background: maroon; float: left;
} #c { background: navy; float: left;
}
```

//实现元素右浮动

```
#c { background: navy; float: right;
}
```

//取消元素的浮动

```
#c { background: navy; float: none;
}
```

刚才的浮动有一个问题：当一个元素盒子被浮动后，下面的元素会自动堆叠处理，导致元素不可见或部分不可见。我们可以使用 clear 属性来处理。

属性	值	说明	CSS 版本
clear	none	允许两边均可浮动	1
left	左边界不得浮动	1	
right	右边界不得浮动	1	
both	两边都不得浮动	1	

//两边均不可浮动

```
#c { background: navy; clear: both;
}
```

第 17 章 CSS 边框与背景[上]

学习要点：

1.声明边框

2.边框样式

3.圆角边框

主讲教师：李炎恢

本章主要探讨 HTML5 中 CSS 边框和背景，通过边框和背景的样式设置，给元素增加更丰富的外观。

一. 声明边框

边框的声明有三个属性设置，样式表如下：

属性	值	说明	CSS 版本
border-width	长度值	设置边框的宽度，可选	1
border-style	样式名称	设置边框的样式，必选	1
border-color	颜色值	设置边框的颜色，可选	1

这三个属性值，只有 border-style 是必须声明，才可以出现边框。而其他两个属性会出现默认值。

//最简单的边框，边框长度默认 3px，边框颜色为黑色

```
div { border-style: solid;
}
```

//配置完整的边框

```
div { border-style: solid; border-width: 2px; border-color: red;
}
```

如果元素长和高均为 200px 时，四个边框均为 2 时，元素的长高总尺寸均为 202px。

二. 边框样式

边框的样式主要有三种，分别是边框长度取值、边框的颜色和边框的线条类型。颜色是通用的颜色代码，和所有其他颜色取值一下。而长度和线条类型，边框有自己独到的部分。

边框宽度取值表如下：

值	说明
长度值	CSS 长度值：比如 px、em 等
百分数	直接设置百分数：1、2、3 等
thin	使用长度名称的预设宽度。这三个值的具体意义由浏览器来定义，从小到大依次增大。
medium	
thick	

一般来说，边框为了更加精准，还要计算元素盒子的总尺寸，使用长度值的比较多。而定义边框线条的样式如下样式表：

值	说明
none	没有边框
dashed	破折线边框
dotted	圆点线边框
double	双线边框
groove	槽线边框
inset	使元素内容具有内嵌效果的边框
outset	使元素内容具有外凸效果的边框
ridge	脊线边框
solid	实线边框

//solid 实线使用频率最高

```
div { border-style: solid; border-width: 10px; border-color: red;
}
```

如果想对四条边中某一条边单独进行设置，可以使用如下样式表：

属性	说明	CSS 版本
border-top-width	定义顶端	1
border-top-style		
border-top-color		
border-middle-width	定义底部	1
border-middle-style		
border-middle-color		
border-left-width	定义左侧	1
border-left-style		
border-left-color		
border-right-width	定义右边	1
border-right-style		
border-right-color		

//只设置顶端

```
div { border-top-style: solid; border-top-width: 10px; border-top-color: red;
}
```

如果四条变都一致，那么没必要分写成三句样式，直接通过简写即可：

属性	值	说明	CSS 版本
border	<宽度> <样式> <颜色>	设置四条边的边框	1
border-top	只设置上边框		
border-middle	只设置下边框		
border-left	只设置左边框		
border-right	只设置右边框		

//简写形式四条边设置

```
div { border: 10px solid red;
}
```

三．圆角边框

CSS3 提供了一个非常实用的圆角边框的设置。使用 border-radius 属性，就可以达到这种效果，样式表如下：

属性	值	说明	CSS 版本
border-radius	长度值或百分数	四条边角	3
border-top-left-radius	长度值或百分数	左上边角	3
border-top-right-radius	长度值或百分数	右上边角	3
border-middle-left-radius	长度值或百分数	左下边角	3
border-middle-right-radius	长度值或百分数	右下边角	3

//设置圆角矩形

```
div { border: 10px solid red; border-radius: 10px;
}
```

//四条边分别设置

```
div { border: 10px solid red; border-radius: 10px 20px 30px 40px;
}
```

第 17 章 CSS 边框与背景[下]

学习要点：

1.设置背景

主讲教师：李炎恢

本章主要探讨 HTML5 中 CSS 边框和背景，通过边框和背景的样式设置，给元素增加更丰富的外观。

一．设置背景

盒模型的尺寸可以通过两种方式实现可见性，一种就是之前的边框，还有一种就是背景。
CSS 背景设置的样式表如下：

属性	说明	CSS 版本
background-color	背景的颜色	1
background-image	背景的图片	1/3
background-repeat	背景图片的样式	1/3
background-size	背景图像的尺寸	3
background-position	背景图像的位置	1
background-attachment	背景图片的滚动	1/3
background-clip	背景图片的裁剪	3
background-origin	背景图片起始点	3
background	背景图片简写方式	1

1.background-color

值	说明	CSS 版本
颜色	设置背景颜色为指定色	1
transparent	设置背景颜色为透明色	1

```
div { background-color: silver;
}
```

解释：设置元素的背景颜色。属性值是颜色值。

```
div b { background-color: transparent;
}
```

解释：默认值为 transparent，为透明的意思。这样<div>内部的元素就会继承<div>的背景色。一般来说这个属性使用频率很低，原因是不需要改变色彩时就默认，需要改变色彩时又是颜色值。

```
body { background-color: silver;
}
```

解释：在<body>元素下可以设置整个页面的背景色。

2.background-image

值	说明	CSS 版本
none	取消背景图片的设置	1
url	通过 URL 设置背景图片	1

```
body { background-image: url(loading.gif);
}
```

解释：url 里面是图片的路径，设置整个页面以这个图片为背景，如果图片不足以覆盖，则复制扩展。

```
div { background-image: none;
}
```

解释：如果多个 div 批量设置了背景，而其中某一个不需要背景，可以单独设置 none 值取消背景。

在 CSS3 中，背景图片还设置了比如线性、放射性等渐变方式。但由于支持度的问题，比如 IE9 尚未支持。我们把这些的新特性放到独立的课程中讲解。

3.background-repeat

值	说明	CSS 版本
repeat-x	水平方向平铺图像	1
repeat-y	垂直方向平铺图像	1
repeat	水平和垂直方向同时平铺图像	1
no-repeat	禁止平铺图像	1

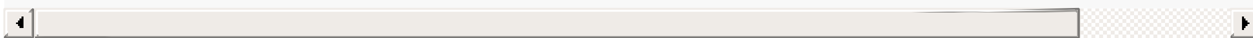
```
body { background-image: url(loading.gif); background-repeat: no-repeat;
}
```

解释：让背景图片只显示一个，不平铺。CSS3 还提供了两个值，由于支持度不佳，这里忽略。

4.background-position

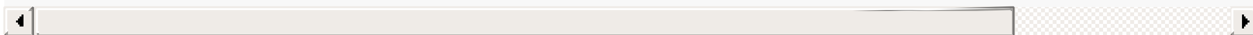
值	说明	CSS 版本
top	将背景图片定位到元素顶部	1
left	将背景图片定位到元素左部	1
right	将背景图片定位到元素右部	1
bottom	将背景图片定位到元素底部	1
center	将背景图片定位到元素中部	1
长度值	使用长度值偏移图片的位置	1
百分数	使用百分数偏移图片的位置	1

```
body { background-image: url(loading.gif); background-repeat: no-repeat; background-posit
}
```



解释：将背景图片置于页面上方，如果想置于左上方则值为：top left。

```
body { background-image: url(loading.gif); background-repeat: no-repeat; background-posit
}
```



解释：使用长度值或百分数，第一值表示左边，第二个值表示上边。

5.background-size

值	说明	CSS 版本
auto	默认值，图像以本尺寸显示	3
cover	等比例缩放图像，使图像至少覆盖容器，但有可能超出容器	3
contain	等比例缩放图像，使其宽度、高度中较大者与容器横向或纵向重合	3
长度值	CSS 长度值，比如 px、em	3
百分数	比如：100%	3

```
body { background-image: url(loading.gif); background-size: cover;
}
```

解释：使用 cover 相当于 100%，全屏铺面一张大图，这个值非常实用。在等比例放大缩小的过程中，可能会有背景超出，当然，这点无伤大雅。

```
div { background-image: url(loading.gif); background-size: contain;
}
```

解释：使用 contain 表示，尽可能让图片完整的显示在元素内。

```
body { background-image: url(loading.gif); background-size: 240px 240px;
}
```

解释：长度值的用法，分别表示长和高。

6.background-attachment

值	说明	CSS 版本
scroll	默认值，背景固定在元素上，不会随着内容一起滚动	1
fixed	背景固定在视窗上，内容滚动时背景不动	1

```
body { background-image: url(loading.gif); background-attachment: fixed;
}
```

解释：fixed 属性会导致背景产生水印效果，拖动滚动条而背景不动。

7.background-origin

值	说明	CSS 版本
border-box	在元素盒子内部绘制背景	3
padding-box	在内边距盒子内部绘制背景	3
content-box	在内容盒子内部绘制背景	3

```
div { width: 400px; height: 300px; border: 10px dashed red; padding: 50px; background-ima
}
```



解释：设置背景起始位置。

8.background-clip

值	说明	CSS 版本
border-box	在元素盒子内部裁剪背景	3
padding-box	在内边距盒子内部裁剪背景	3
content-box	在内容黑子内部裁剪背景	3

```
div { width: 400px; height: 300px; border: 10px dashed red; padding: 50px; background-ima
}
```



解释：在内边距盒子内部裁剪背景。

9.background

```
div { width: 400px; height: 300px; border: 10px dashed red; padding: 50px; background: si
}
```



解释：完整的简写顺序如下：

[background-color]

[background-image]

[background-repeat]

[background-attachment]

[background-position]/[background-size]

[background-origin]

[background-clip]

第 18 章 CSS 表格与列表

学习要点：

1.表格样式

2.列表样式

3.其他功能

主讲教师：李炎恢

本章主要探讨 HTML5 中 CSS 表格和列表，通过表格和列表的样式设置，让表格和列表显示更加多元化。

一．表格样式

表格有五种独有样式，样式表如下：

属性	值	说明	CSS 版本
border-collapse	边框样式	相邻单元格的边框样式	2
border-spacing	长度值	相邻单元格边框的间距	2
caption-side	位置名称	表格标题的位置	2
empty-cells	显示值	空单元格是否显示边框	2
table-layout	布局样式	指定表格的布局样式	2

1.border-collapse

值	说明	CSS 版本
separate	默认值，单元格边框独立	2
collapse	单元格相邻边框被合并	2

```
table { border-collapse: collapse;
}
```

解释：单元格相邻的边框被合并。

2.border-spacing

值	说明	CSS 版本
长度值	0 表示间距，其他使用 CSS 长度值	2

```
table { border-spacing: 10px;
}
```

解释：border-collapse: separate;的状态下才有效。因为要设置间距，不能合并。

3.caption-side

值	说明	CSS 版本
top	默认值，标题在上方	2
bottom	标题在下方	2

```
table { table-layout: fixed;
}
```

解释：内容过长后，不会拉伸整个单元格。

4.empty-cells

值	说明	CSS 版本
show	默认值，显示边框	2
hide	不显示边框	2

```
table { empty-cells: hide;
}
```

解释：单元格内容为空是隐藏边框。

5.table-layout

值	说明	CSS 版本
auto	默认值，内容过长时，拉伸整个单元格	2
fixed	内容过长时，不拉伸整个单元格	2

```
table {    table-layout: fixed;

}
```

解释：内容过长后，不会拉伸整个单元格。

二．列表样式

列表有四种独有样式，样式表如下：

属性	值	说明	CSS 版本
list-style-type	类型值	列表中的标记类型	1/2
list-style-image	none 或 url	图像作为列表标记	1
list-style-position	位置值	排列的相对位置	1
list-style	简写属性	列表的简写形式	1

1.list-style-type

值	说明	CSS 版本
none	没有标记	1
disc	实心圆	1
circle	空心圆	1
square	实心方块	1
decimal	阿拉伯数字	1
lower-roman	小写罗马数字	1
upper-roman	大写罗马数字	1
lower-alpha	小写英文字母	1
upper-roman	大写英文字母	1

```
ul { list-style-type: square;
}
```

解释：列表前缀的标记类型，这里是 CSS1 版本的。CSS2 版本还包含比如日文、亚美尼亚数字、希腊文等前缀。有兴趣的可以参考 CSS 手册。

2.list-type-position

值	说明	CSS 版本
outside	默认值，标记位于内容框外部	1
inside	标记位于内容框内部	1

```
ul { width: 120px; list-style-position: inside;
}
```

解释：标记位于内容框的内部。

3.list-type-image

值	说明	CSS 版本
none	不使用图像	1
url	通过 url 使用图像	1

```
ul { list-style-image: url(bullet.png);
}
```

解释：使用图片作为前缀的标记。

4.list-style

```
ul { list-style: lower-alpha inside url(bullet.png);
}
```

解释：简写形式。

三．其他功能

在<table>中<td>单元格，我们可以使用 text-align 属性水平对齐，但是垂直对齐就无法操作了。CSS 提供了 vertical-align 属性用于垂直对齐。

值	说明	CSS 版本
baseline	内容对象与基线对齐	1
top	内容对象与顶端对齐	1
middle	内容对象与中部对齐	1
bottom	内容对象与底部对齐	1

```
table tr td { vertical-align: bottom;
}
```

解释：将单元格内的内容对象实现垂直对齐。

值	说明	CSS 版本
sub	垂直对齐文本的下标	1
super	垂直对齐文本的上标	1

```
b { vertical-align: super;
}
```

解释：文本上下标设置。

值	说明	CSS 版本
长度值	设置上下的偏移量，可以是负值	1
百分比	同上	1

```
div span { vertical-align: -200px;
}
```

解释：负值往下，正值往上。如果默认基线在上面，用负数。如果默认基线在下面，用正值。

第 19 章 CSS 其他样式

学习要点：

1.颜色和透明度

2.盒子阴影和轮廓

3.光标样式

主讲教师：李炎恢

本章主要探讨 HTML5 中 CSS 其他剩下几个常用的样式，包括颜色、透明度、盒子的阴影轮廓以及光标的样式。

一．颜色和透明度

颜色我们之前其实已经用的很多了，比如字体颜色、背景颜色、边框颜色。但除了背景颜色和边框颜色讲解过，字体颜色却没有系统的讲解过。设置字体颜色其实也成为文本块的前景色。

属性	值	说明	CSS 版本
color	颜色值	设置文本前景色	1

```
p { color: red;
}
```

解释：设置文本颜色。

CSS3 提供了一个属性 opacity，可以设置元素的透明度。

属性	值	说明	CSS 版本
opacity	0 ~ 1	设置元素的透明度	3

```
p { color: red; opacity: 0.5;
}
```

解释：设置元素的透明度。

二．盒子阴影和轮廓

1.box-shadow

CSS3 提供了一个非常实用的效果样式，就是阴影效果。通过 box-shadow 属性来实现，样式表如下：

属性	值	说明	CSS 版本
box-shadow	hoffset	阴影的水平偏移量，是一个长度值，正值表示阴影向右偏移，负值表示阴影向左偏移。	3
	voffset	阴影的垂直偏移量，是一个长度值，正值代表阴影位于元素盒子的下方，负值代表阴影位于元素盒子上方。	3
	blur	（可选）指定模糊值，是一个长度值，值越大盒子的边界越模糊。默认值为 0，边界清晰	3
	spread	（可选）指定阴影延伸半径，是一个长度值，正值代表阴影向盒子各个方向延伸扩大，负值代表阴影沿相反方向缩小	3
	color	（可选）设置阴影的颜色，如果省略，浏览器会自行选择一个颜色	3
	inset	（可选）将外部阴影设置为内部阴影。	3

```
div { width: 200px; height: 200px; border: 10px solid silver; box-shadow: 5px 4px 10px 2p
}
```

解释：给元素盒子增加阴影效果。

```
box-shadow: 5px 4px 10px 2px gray inset;
```

解释：实现内部阴影。

2.outline

CSS3 还提供了轮廓样式，它和边框一样，只不过它可以在边框的外围再加一层。样式表如下：

属性	值	说明	CSS 版本
outline-color	颜色	外围轮廓的颜色	3
outline-offset	长度	轮廓距离元素边框边缘的偏移量	3
outline-style	样式	轮廓样式，和 border-style 一致	3
ontline-witdh	长度	轮廓宽度	3
outline	简写	<颜色> <样式> <宽度>	3

```
div { width: 200px; height: 200px; border: 10px solid silver; outline: 10px double red;
}
```

解释：在边框的外围再增加一圈轮廓。

三．光标样式

我们不但可以指定页面上的元素样式，就连光标的样式也可以指定。样式表如下：

属性	值	说明	CSS 版本
cursor	光标样式	auto,default,none,context-menu,help,pointer,progress,wait,cell,crosshair,text,vertical-text,alias,copy,move,no-drop,not-allowed,e-resize,n-resize,ne-resize,nw-resize,s-resize,se-resize,sw-resize,w-resize,ew-resize,ns-resize,nesw-resize,nwse-resize,col-resize,row-resize,all-scroll	1

```
div { cursor: move;
}
```

解释：设置当前元素的光标为移动光标。

第 20 章 CSS3 前缀和 rem

学习要点：

1.CSS3 前缀

2.长度单位 rem

主讲教师：李炎恢

本章主要探讨 HTML5 中 CSS 在发展中实行标准化的一些问题，重点探讨 CSS3 中新属性前缀问题和新的单位 rem。

一．**CSS3** 前缀

在 CSS3 的很多新属性推出时，这些属性还处在不太稳定的阶段，随时可能被剔除。而此时的浏览器厂商为了实现这些属性，采用前缀方法。各大厂商前缀列表如下：

浏览器	厂商前缀
Chrome、Safari	-webkit-
Opera	-o-
Firefox	-moz-
Internet Explorer	-ms-

我们之前学习过几个 CSS3 的新属性，比如：box-shadow、border-radius、opacity等。这几个属性我们在前面的使用中，并没有添加所谓的浏览器厂商前缀。那是因为，这些属性已经在主流浏览器或版本成为了标准。具体进化步骤如下：

1.属性尚未提出，这个属性所有浏览器不可用；

2.属性被提出，但未列入标准，浏览器厂商通过私有前缀来支持该属性；

3.属性被列入标准，可以使用前缀或不使用前缀来实现属性特性；

4.属性不需要再使用前缀，所有浏览器都稳定支持。

我们就拿 border-radius 举例，它是 CSS3 的标准属性。早期的时候处于实验阶段，尚未列入标准时，需要使用厂商前缀。具体浏览器支持度如下：

属性	浏览器	带前缀版本	不带前缀版本	标准/实验
border-radius	IE	不支持	9.0+	标准
Firefox	3.0 需带-moz-	4.0+		
Safari	3.1 需带-webkit-	5.1+		
Chrome	4.0	5.0+		
Opera	不支持	10.5+		

如果是手机等移动端一般采用的是 IOS 或安卓系统，那么基本上它的引擎是 webkit，直接参考-webkit-即可。

在 CSS3 手册上，Chrome 支持 border-radius 的版本为 13.0，而部分教材和文章上写到只要 5.0。当然，这里可能表达的含义可能不同。而截至到 2015 年 4 月份最新的 Chrome 版本已经到 41.0 了，所以，不管是 5.0 还是 13.0 都是老古董了，没必要深究。Opera 支持 border-radius 版本为 11.5，而目前的版本是 28.0，也无伤大雅了。

而被列入标准的 box-shadow 和 opacity 基本与 border-radius 前缀版本一致。

//因为目前处在标准阶段，没必要写前缀了

```
div { border-radius: 50px;
}
```

//实验阶段的写法

```
div { -webkit-border-radius: 50px; -moz-border-radius: 50px; border-radius: 50px;
}
```

实验阶段的写法有三句，分别解释一下：-webkit-表示 Chrome 浏览器的私有属性前缀、-moz-表示 Firefox 私有属性前缀，如果是处于实验阶段的旧版本浏览器，那么不支持 border-radius，从而通过厂商前缀的方式来支持。如果是新版浏览器，已经是处于标准阶段，那么又有两种说法：1.如果新版浏览器废弃了前缀，那么前缀写法就不支持了，仅支持标准写法；2.如果新版浏览器没有废弃前缀，那么两种写法都可以，但要注意顺序，且属性值要保持一致。

如果同时出现四个前缀+一个标准写法，四个前缀是当实验阶段时让四种引擎的浏览器厂商支持自己的私有属性，一个标准写法表示当这个属性列入标准后，使用新版浏览器运行时直接执行这个标准属性。

//前缀写法写在标准后面，且值不一样，就会出现问题

```
div { border-radius: 50px; -webkit-border-radius: 100px;
}
```

特别注意：1.IE 的前缀-ms-，和 Opera 的前缀-o-，在 border-radius 中不存在；2.现在的 Opera 浏览器也支持-webkit-前缀，-webkit-border-radius 就能支持；3.Safari for Windows 已被苹果公司在 2012 年放弃，遗留版本为 5.1.7。

最后说明：W3C 官方的立场表示实验阶段的属性仅为了测试，未来有可能修改或删除。而大量的开发者也认为在实际项目中不应该使用前缀，而使用一种优雅降级保证一定的用户体验，而通过渐进增减的方式让新版高级浏览器保证最高的效果。

二．长度单位 **rem**

CSS3 引入了一些新的尺寸单位，这里重点推荐一个：rem 或者成为(根 em)。目前主流的现代浏览器都很稳定的支持。它和 em、百分比不同的是，它不是与父元素挂钩，而是相对于根元素<html>的文本大小来计算的，这样能更好的统一整体页面的风格。

//首先，来一段 HTML

```
<h1>标题<em>小标题</em></h1>
<p> 我是一个段落，我是一段<code>代码</code>
</p>
```

//其次来一段 CSS

```
html { font-size: 62.5%;
} h1 { font-size: 3em;
} p { font-size: 1.4em;
}
```

这里做几个解释，我们在之前的 Web 设计中大量使用了 px 单位进行布局。因为，早期的固定布局使用 px 较为方便，逐渐养成了这种习惯。而使用 em 单位其实更加灵活，尤其是在修改样式时，只需要修改一下挂钩元素的那个大小即可，无须每个元素一个个修改。

但就算是 em，还是有一定问题。网页默认的字号大小为 16px，然后通过<html>设置 62.5%，将网页基准设置为 10px。而<h1>设置为 3em，就是自身大小的 3 倍；<p>设置为 1.4em，就是 10px 的 1.4 倍，即 14px。

现在问题来了，<code>里面的文本想设置 11px，怎么办呢？设置 1.1em 吗？不对，因为它挂钩的父元素不是<html>而是<p>变成了 14px 的 1.1 倍了，而想设置 11px，则需要设置 0.786 倍才行。但是，这样的计算量太大了。所以，W3C 推出了直接基于根元素单位：rem。

//直接基于<html>的单位

```
code { font-size: 1.1 rem;
}
```

浏览器	**rem 单位**
Opera	11.6+
Firefox	3.6+
Safari	5.0+
Chrome	6.0+
IE	9.0+

第 21 章 CSS3 文本效果

学习要点：

1.文本阴影

2.文本裁剪

3.文本描边

4.文本填充

主讲教师：李炎恢

本章主要探讨 HTML5 中 CSS3 中文本效果，其中也包含一些之前讲过的 CSS3 文本属性。

一．文本阴影

CSS3 提供了 text-shadow 文本阴影效果，这个属性在之前讲过，只是没有涉及浏览器支持情况。

text-shadow	Opera	Firefox	Chrome	IE	Safari
9.5+	3.5+	4+	10+	3.1+	

这里有几个注意点：1.text-shadow 在 CSS2 的时候出现过，但各大浏览器碍于消耗大量的资源，迟迟未支持，然后在 CSS2.1 中剔除了。现在，CSS3 已经全面支持了；2.最低支持版本上，不同手册和教材上都不太同，但版本年代久远，无伤大雅。最准确的可以查询这个网站：http://caniuse.com。最需要注意的只有 IE10 才支持，IE9 不支持的；3.目前的浏览器不需要给这个属性加上任何前缀，具体查询前缀版本可以访问刚才提供的地址。

//正值阴影

```
text-shadow: 1px 1px 1px red;
```

//负值阴影

```
text-shadow: -1px -1px 1px red;
```

//多重阴影叠加

```
text-shadow:0px 0px 0 rgb(188,178,188),
            1px 0px 0 rgb(173,163,173),
            2px 0px 0 rgb(157,147,157),
            3px 0px 0 rgb(142,132,142),
            4px 0px 0 rgb(126,116,126),
            5px 0px 0 rgb(111,101,111),
            6px 0px 0 rgb(95,85,95),
            7px 0px 0 rgb(79,69,79),
            8px 0px 7px rgba(0,0,0,0.35),
            8px 0px 1px rgba(0,0,0,0.5),
            0px 0px 7px rgba(0,0,0,0.2);
```

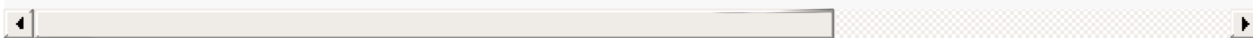
二．文本裁剪

CSS3 提供了 text-overflow 属性来控制文本的溢出部分，它的作用是对溢出的部分裁剪掉，然后判定是否添加省略号。首先我们先看下样式表的属性，如下：

属性值	说明
clip	默认值，裁剪文本时不添加"..."省略号
ellipsis	裁剪文本时添加"..."省略号

//必须不换号和使用 overflow 控制溢出

```
p { width: 160px; white-space: nowrap; background: silver; /*text-overflow: clip;*/ text-
}
```



对于 text-overflow 的支持度，是根据它的属性值来判定的，不同的属性值浏览器支持度不同。

属性值	Opera	Firefox	Chrome	IE	Safari
clip	9.63+	2.0+	1.0+	6.0+	3.1+
ellipsis	10.5+	6.0+	1.0+	6.0+	3.1+

//在 Opera 早期版本 10.0 ~ 10.1 中，需要使用带前缀的-o-。

```
p { -o-text-overflow: ellipsis; text-overflow: ellipsis;
}
```

而在 Opera 主流版本中，引擎已经靠拢 webkit，则不需要-o-了。目前来说，也不需要兼容-o-了。

三．文本描边

CSS3 提供了描边属性，即 text-stroke、text-stroke-width、text-stroke-color。目前只有 webkit 引擎的浏览器支持，并且必须加上-webkit-前缀才能有效。

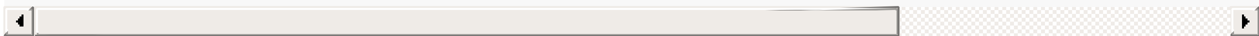
属性	Opera	Firefox	Chrome	IE	Safari
text-stroke 系列	15.0+	不支持	4.0+	不支持	3.1+

//实验阶段的产物，了解即可

```
p { font-size: 50px; -webkit-text-stroke: 1px red;
}
```

//修改描边的颜色和厚度

```
p { font-size: 50px; -webkit-text-stroke: 1px red; -webkit-text-stroke-color: orange; -we
}
```



四·文本填充

CSS3 提供了一个文本颜色填充功能：text-fill-color，感觉和 color 属性很像。其实在配合其他属性才能达到不一样的效果。

属性	Opera	Firefox	Chrome	IE	Safari
text-fill-color	15.0+	不支持	4.0+	不支持	3.1+

//不配合背景样式时，和 color 属性没区别

```
p { font-size: 150px; -webkit-text-fill-color: red;
}
```

//和 CSS3 背景的新特性搭配产生渐变文字

```
p { font-size: 150px; font-family: 黑体; background: -webkit-linear-gradient(top,#eee,#aaa
}
```



第 21 章 CSS3 文本效果

学习要点：

1.文本阴影

2.文本裁剪

3.文本描边

4.文本填充

主讲教师：李炎恢

本章主要探讨 HTML5 中 CSS3 中文本效果，其中也包含一些之前讲过的 CSS3 文本属性。

一．文本阴影

CSS3 提供了 text-shadow 文本阴影效果，这个属性在之前讲过，只是没有涉及浏览器支持情况。

text-shadow	Opera	Firefox	Chrome	IE	Safari
9.5+	3.5+	4+	10+	3.1+	

这里有几个注意点：1.text-shadow 在 CSS2 的时候出现过，但各大浏览器碍于消耗大量的资源，迟迟未支持，然后在 CSS2.1 中剔除了。现在，CSS3 已经全面支持了；2.最低支持版本上，不同手册和教材上都不太同，但版本年代久远，无伤大雅。最准确的可以查询这个网站：http://caniuse.com。最需要注意的只有 IE10 才支持，IE9 不支持的；3.目前的浏览器不需要给这个属性加上任何前缀，具体查询前缀版本可以访问刚才提供的地址。

//正值阴影

```
text-shadow: 1px 1px 1px red;
```

//负值阴影

```
text-shadow: -1px -1px 1px red;
```

//多重阴影叠加

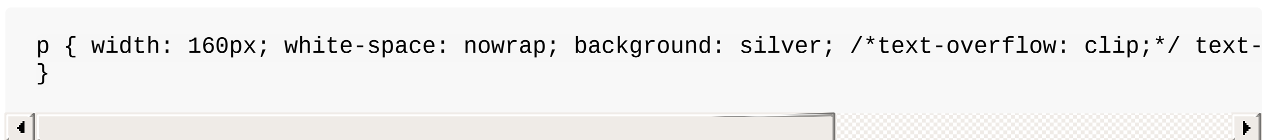
```
text-shadow:0px 0px 0 rgb(188,178,188),
            1px 0px 0 rgb(173,163,173),
            2px 0px 0 rgb(157,147,157),
            3px 0px 0 rgb(142,132,142),
            4px 0px 0 rgb(126,116,126),
            5px 0px 0 rgb(111,101,111),
            6px 0px 0 rgb(95,85,95),
            7px 0px 0 rgb(79,69,79),
            8px 0px 7px rgba(0,0,0,0.35),
            8px 0px 1px rgba(0,0,0,0.5),
            0px 0px 7px rgba(0,0,0,0.2);
```

二·文本裁剪

CSS3 提供了 text-overflow 属性来控制文本的溢出部分，它的作用是对溢出的部分裁剪掉，然后判定是否添加省略号。首先我们先看下样式表的属性，如下：

属性值	说明
clip	默认值，裁剪文本时不添加"..."省略号
ellipsis	裁剪文本时添加"..."省略号

//必须不换号和使用 overflow 控制溢出



```
p { width: 160px; white-space: nowrap; background: silver; /*text-overflow: clip;*/ text-
}
```

对于 text-overflow 的支持度，是根据它的属性值来判定的，不同的属性值浏览器支持度不同。

属性值	Opera	Firefox	Chrome	IE	Safari
clip	9.63+	2.0+	1.0+	6.0+	3.1+
ellipsis	10.5+	6.0+	1.0+	6.0+	3.1+

//在 Opera 早期版本 10.0 ~ 10.1 中，需要使用带前缀的-o-。

```
p { -o-text-overflow: ellipsis; text-overflow: ellipsis;
}
```

而在 Opera 主流版本中，引擎已经靠拢 webkit，则不需要-o-了。目前来说，也不需要兼容-o-了。

三·文本描边

CSS3 提供了描边属性，即 text-stroke、text-stroke-width、text-stroke-color。目前只有 webkit 引擎的浏览器支持，并且必须加上-webkit-前缀才能有效。

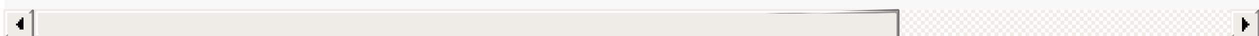
属性	Opera	Firefox	Chrome	IE	Safari
text-stroke 系列	15.0+	不支持	4.0+	不支持	3.1+

//实验阶段的产物，了解即可

```
p { font-size: 50px; -webkit-text-stroke: 1px red;
}
```

//修改描边的颜色和厚度

```
p { font-size: 50px; -webkit-text-stroke: 1px red; -webkit-text-stroke-color: orange; -we
}
```



四．文本填充

CSS3 提供了一个文本颜色填充功能：text-fill-color，感觉和 color 属性很像。其实在配合其他属性才能达到不一样的效果。

属性	Opera	Firefox	Chrome	IE	Safari
text-fill-color	15.0+	不支持	4.0+	不支持	3.1+

//不配合背景样式时，和 color 属性没区别

```
p { font-size: 150px; -webkit-text-fill-color: red;
}
```

//和 CSS3 背景的新特性搭配产生渐变文字

```
p { font-size: 150px; font-family: 黑体; background: -webkit-linear-gradient(top,#eee,#aaa
}
```



第 23 章 CSS3 边框图片效果

学习要点：

1.属性初探

2.属性解释

3.简写和版本

主讲教师：李炎恢

本章主要探讨 HTML5 中 CSS3 中边框图片背景的效果，通过这个新属性让边框更加的丰富多彩。

一．属性解释

CSS3 提供了一个新的属性集合，用这几个属性可以嵌入图片形式的边框。这样，边框就可以自定义了。

1.border-image-source //引入背景图片地址

2.border-image-slice //切割引入背景图片

3.border-image-width //边框图片的宽度

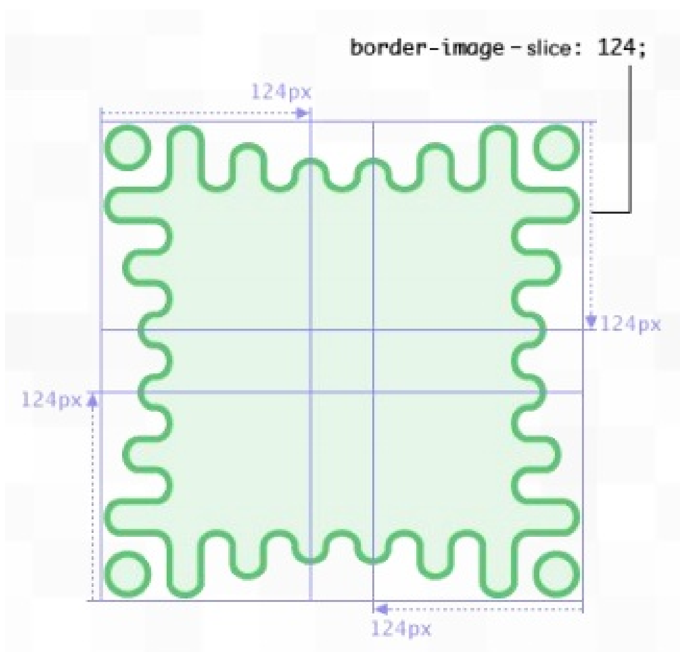
4.border-image-repeat //边框背景图片的排列方式

5.border-image-outset //边框背景向外扩张

6.border-image //上面五个属性的简写方式

二．属性解释

要实现边框背景，我们对图片也有一定的要求，否则效果不能完全体现出来。图片可以通过九宫格的切分来了解它。我们使用 W3C 官网上教学的图片来讲解一下。



如上图所示，九宫格并不一定要求每一个格子大小都相同。当然，如果相同的话，制作边框背景就相对容易一点。比如如下这张图片：



首先，用 Photoshop 软件分析一下这个标准九宫格的总体大小和每个格子的大小。最终得出图片总大小为 81px 正方形，四个角的大小为 27px 的正方形，其余五个角也是 27px。

那么，第一步：先创建一个盒子区域，大小为 400x400 的矩形。然后设置引入边框图像。

//引入边框图像

```
border-image-source: url(border.png);
```

单单只有这句话，webkit 引擎下的浏览器会在盒子区块的四个角看到一丁点图像的影子。而其他浏览器什么都看不到。这是由于没有设置边框背景图像的宽度导致的。

//设置边框图像宽度，上右下左，可以设置四个值

```
border-image-width: 81px;
```

这里设置的是边框图像的宽度，而不是边框宽度。当你设置边框宽度，你会发现，文本会偏移。而边框图像的宽度不会挤压文本。

//设置边框的宽度

```
border-width: 20px;
```

以上设置完毕后，支持边框背景图片的浏览器会在四个角落铺上这张图片的完整形式。这个时候需要通过引入切割属性来配置背景图片的显示方式。

//首先，边框图像宽度设置为 27px 和一个单格宽高一致

```
border-image-width: 27px;
```

//设置切割属性的大小

```
border-image-slice: 27;
```

这里的 27 不需要设置 px 像素，因为它默认就是像素。设置 27 之后，我们会发现边框的四个角正好是橘红色的四个角。那么你可以逐步放大或逐步放下这个值，来体验一下它的变化。

//从 27 逐步放大到 81，四个角都慢慢缩小，各自显示一个完整的图像

```
border-image-slice: 81;
```

//从 27 逐步缩小到 0，发现四个角都慢慢变大，配合 fill 整体显示一个完整图像

```
border-image-slice: 0 fill;
```

上面只是单独设置了一个像素表示四个边切割的大小，你也可以设置百分比、浮点值或者分别设置四个变的大小。

//33.5%差不多 27

```
border-image-slice: 33.5%;
```

//上下设置 27，左右设置 0

```
border-image-slice: 27 0;
```

如果想让边框背景向外扩张，那么可以进行扩张设置。

//向外扩张 20px，也可以是浮点值，比如 2.2

```
border-image-outset: 20px;
```

四个角设定好之后，我们要设定四个变的显示排列方式。使用 border-image-repeat 属性，有四个值提供使用，分别如下表：

属性	说明
stretch	指定用拉伸方式填充边框背景图。默认值。
repeat	指定用平铺方式来填充边框背景图。当图片碰到边界时，如果超过则被截断。
round	指定用平铺方式来填充边框背景图。图片会根据边框的尺寸动态调整图片的大小，直至正好可以铺满整个边框。
space	指定用平铺方式来填充边框背景图。图片会根据边框的尺寸动态调整图片的之间的间距，直至正好可以铺满整个边框。

//拉伸方式填充，当然，通过上右下左设置四个边均可

```
border-image-repeat: stretch;
```

//平铺填充，超过则被截断

```
border-image-repeat: repeat;
```

//平铺填充，动态调整图片大小直至铺满

```
border-image-repeat: round;
```

//平铺填充，动态调整图片的间距直至铺满

```
border-image-repeat: space;
```

//另一个按钮的小例子

```
div { width: 400px; height: 40px; border-image-source: url(button.png); border-image-widt
}
```



三．简写和版本

//border-image 简写格式很简单，具体如下：

```
border-image：<' border-image-source '> || <' border-image-slice '> [ /<' border-image-wi
```



//以上是手册上摘录的，转换成实际格式如下：

```
border-image: url(border.png) 27/27px round;
```

对于支持的浏览器及版本如下表：

	Opera	Firefox	Chrome	Safari	IE
部分支持需带前缀	11.5~12.1	3.5 ~ 14	4 ~ 14	3.1 ~ 5.1	无
支持需带前缀	无	无	无	无	无
支持不带前缀	15+	15+	15+	6+	11.0+

//兼容加上前缀

```
-webkit-border-image: url(border.png) 27/27px round;
-moz-border-image: url(border.png) 27/27px round;
-o-border-image: url(border.png) 27/27px round;
border-image: url(border.png) 27/27px round;
```

第 24 章 CSS3 变形效果[下]

学习要点：

1.3D 变形简介

2.transform-style

3.perspective

4.3D 变形属性

主讲教师：李炎恢

本章主要探讨 HTML5 中 CSS3 的变形效果，主要接着上节课的 2D 平面变形转换到 3D 立体变形。

一 · **3D** 变形简介

之前我们学习了元素的平移、旋转、缩放和倾斜等功能。这些效果只是单纯在二维平面图上的，我们称之为 2D。那么其实 CSS3 也提供了三维立体的一些功能效果，并且目前较新的主流浏览器都比较支持，只不过比 2D 晚一些，对浏览器的版本要求也要高一些。

由于 3D 是立体三维，在 x、y 轴的基础上一般会多出一个 z 轴，深入跃出轴。以下是 3D 变形的属性值表，如下：

属性值	说明
translate3d(x,y,z)	3D 方式平移元素，设置 x、y 和 z 轴
translateZ(z)	设置 3D 方式平移元素的 z 轴
scale3d(x,y,z)	3D 方式缩放一个元素
scaleZ(z)	设置 3D 方式缩放元素的 z 轴
rotate3d(x,y,z,a)	3d 方式旋转元素
rotateX(a)	分别设置 3D 方式的旋转元素的 x、y 和 z 轴
rotateY(a)	
rotateZ(a)	
perspective(长度值)	设置一个透视投影矩阵
matrix3d(多个值)	定义一个矩阵

3D 变形比 2D 变形出来的要晚一些，所以如果需要兼容旧版本浏览器，可以对照这个表。具体如下：

	Opera	Firefox	Chrome	Safari	IE
支持需带前缀	15 ~ 22	10 ~ 15	12 ~ 35	4 ~ 8	无
支持不带前缀	23+	16+	26+	无	10.0+

//兼容版本完整形式

```
-webkit-transform: translateZ(200px);
-moz-transform: translateZ(200px);
-o-transform: translateZ(200px);
-ms-transform: translateZ(200px);
transform: translateZ(200px);
```

二．**transform-style**

transform-style 属性是指定嵌套元素如何在 3D 空间中呈现。

属性值	说明
flat	默认值，表示所有子元素在 2D 平面呈现。
preserve-3d	表示子元素在 3D 空间中呈现。

//一般设置到当前元素的父元素

```
transform-style: preserve-3d;
```

需要再配合后面的功能属性和变形配置，才能看到效果。同样，这个属性也需要加上各种厂商前缀。

三．**perspective**

perspective 是 3D 变形的重要属性，该属性会设置查看者的位置，并将可视内容映射到一个视锥上，继而投放到一个 2D 平面上。

属性值	说明
none	默认值，表示无限的角度来看 3D 物体，但看上去是平的。
长度值	接受一个长度单位大于 0 的值，其单位不能为百分比。值越大，角度出现的越远，就好比你人离远一点看物体。值越小，正相反。

//设置查看者的距离位置，一般设置在元素的父元素上

```
perspective: 1000px;
```

需要再配合后面的功能属性和变形配置，才能看到效果。同样，这个属性也需要加上各种厂商前缀。

四．**3D** 变形属性

我们运用前面 3D 功能属性 transform-style 和 perspective 来构建 3D 变形效果。

1.translate3d(x,y,z)

//需要 3D 位移的 HTML 结构，必须有父元素包含

```
<div id="a">
    <img src="img.png" alt="" />
</div>
```

//CSS 部分，父元素设置 3D 呈现且设置透视距离

```
#a { perspective: 1000px; transform-style: preserve-3d;
} img {
    /*z 轴可以是负值*/ transform: translate3d(300px,100px,240px);
}
```

2.translateZ(z)

//可以单独设置 z 轴，z 轴可以是负值

```
img { transform: translateZ(240px);
}
```

3.scale3d(x,y,z)

//3D 缩放，单独设置无效，需要配合角度

```
img { transform: scale3d(1,1,1.5) rotateX(45deg);
}
```

4.scaleZ(z)

//单独设置 z 轴，x 和 y 轴默认为 1

```
img { transform: scaleZ(1.5) rotateX(45deg);
}
```

5.rotate3d(x,y,z,a)

//设置 3D 旋转，a 表示角度，xyz 是 0 或 1 之间的数值

```
transform: rotate3d(1,0,0,45deg);
```

6.rotateX(a)**、rotateY(a)、**rotateZ(a)

//单独设置 3D 旋转

```
transform: rotateX(45deg);
transform: rotateY(45deg);
transform: rotateZ(45deg);
transform: rotateX(45deg) rotateY(45deg) rotateZ(45deg);
```

最后一个 matrix3d 就不多说了，忽略。

CSS3 还提供了 perspective-origin 属性来设置 3D 变形中的源点角度。该属性默认值为 50% 50%也就是 center center。

属性值	说明
百分数值	指定元素 x 轴或 y 轴的起点
长度值	指定距离
left	指定 x 轴的位置
center	
right	
top	指定 y 轴的位置
center	
bottom	

//源点设置为右上方变形

```
perspective-origin: top right;
```

CSS3 还提供了一个在元素中设置透视的值 perspective(长度值)，但它还是和在父元素设置有一定不同。因为父元素整个作为透视，而元素自己作为透视，导致不同。

//具体测试看透视的距离

```
img { transform: perspective(1000px) rotateY(45deg);
}
```

第 25 章 CSS3 过渡效果

学习要点：

1.过渡简介

2.transition-property

3.transition-duration

4.transition-timing-function

5.transition-delay

6.简写和版本

主讲教师：李炎恢

本章主要探讨 HTML5 中 CSS3 的过渡效果，通过这个功能可以不借助 JavaScript 来实现简单的用户交互功能。

一．过渡简介

过渡效果一般是通过一些简单的 CSS 动作触发平滑过渡功能，比如：:hover、:focus、:active、:checked 等。CSS3 提供了 transition 属性来实现这个过渡功能，主要属性如下表：

属性	说明
transition-property	指定过渡或动态模拟的 CSS 属性
transition-duration	指定完成过渡所需的时间
transition-timing-function	指定过渡的函数
transition-delay	指定过渡开始出现的延迟时间
transition	简写形式，按照上门四个属性值连写

我们先创建一个没有过渡效果的元素，然后通过:hover 来触发它。在没有任何过渡效果的触发，会立即生硬的执行触发。

//设置元素样式

```
div { width: 200px; height: 200px; border: 1px solid green;
}
```

//鼠标悬停后背景变黑，文字变白

```
div:hover { background-color: black; color: white; margin-left: 50px;
}
```

二 · **transition-property**

首先，设置过渡的第一个属性就是指定过渡的属性。同样，你需要指定你要过渡某个元素的两套样式用于用户和页面的交互。那么就使用 transition-property 属性，详细属性值如下表：

属性值	说明
none	没有指定任何样式
all	默认值，指定元素所支持 transition-property 属性的样式
指定样式	指定支持 transition-property 的样式

从上门的列表中来看，一般来说，none 用于本身有过渡样式从而取消。而 all，则是支持所有 transition-property 样式，还有一种是指定 transition-property 中的某些样式。那么 transition-proerty 支持的样式有哪些？如下表所示：

样式名称	样式类型
background-color	color(颜色)
background-image	only gradients(渐变)
background-position	percentage, length(百分比，长度值)
border-bottom-color	color
border-bottom-width	length
border-color	color
border-left-color	color
border-left-width	length
border-right-color	color
border-right-width	length
border-spacing	length
border-top-color	color
border-top-width	length
border-width	length
bottom	length, percentage
color	color
crop	rectangle
font-size	length, percentage

font-weight	number
grid-*	various
height	length, percentage
left	length, percentage
letter-spacing	length
line-height	number, length, percentage
margin-bottom	length
margin-left	length
margin-right	length
margin-top	length
max-height	length, percentage
max-width	length, percentage
min-height	length, percentage
min-width	length, percentage
opacity	number
outline-color	color
outline-offset	integer
outline-width	length
padding-bottom	length
padding-left	length
padding-right	length
padding-top	length
right	length, percentage
text-indent	length, percentage
text-shadow	shadow
top	length, percentage
vertical-align	keywords, length, percentage
visibility	visibility
width	length, percentage
word-spacing	length, percentage
z-index	integer

zoom	number

//设置背景和文字颜色采用过渡效果

```
transition-property: background-color, color, margin-left;
```

三．**transition-duration**

如果单纯设置过渡的样式，还不能够立刻实现效果。必须加上过渡所需的时间，因为默认情况下过渡时间为 0。

//设置过渡时间为 1 秒钟，如果是半秒钟可以设置为.5s

```
transition-duration: 1s;
```

四．**transition-timing-function**

当过渡效果运行时，比如产生缓动效果。默认情况下的缓动是：元素样式从初始状态过渡到终止状态时速度由快到慢，逐渐变慢，即 ease。也是默认值，其他几种缓动方式如下表所示：

属性值	说明
ease	默认值，元素样式从初始状态过渡到终止状态时速度由快到慢，逐渐变慢。等同于贝塞尔曲线(0.25, 0.1, 0.25, 1.0)
linear	元素样式从初始状态过渡到终止状态速度是恒速。等同于贝塞尔曲线(0.0, 0.0, 1.0, 1.0)
ease-in	元素样式从初始状态过渡到终止状态时，速度越来越快，呈一种加速状态。等同于贝塞尔曲线(0.42, 0, 1.0, 1.0)
ease-out	元素样式从初始状态过渡到终止状态时，速度越来越慢，呈一种减速状态。等同于贝塞尔曲线(0, 0, 0.58, 1.0)
ease-in-out	元素样式从初始状态过渡到终止状态时，先加速，再减速。等同于贝塞尔曲线(0.42, 0, 0.58, 1.0)

//恒定速度

```
transition-timing-function: linear;
```

以上五种都是设定好的属性值，我们也可以自定义这个缓动。使用 cubic-bezier() 属性值，里面传递四个参数 p0,p1,p2,p3，值在 0~1 之间。

//自定义缓动

```
transition-timing-function: cubic-bezier(0.25, 0.67, 0.11, 0.55);
```

至于具体这些数值干什么的，怎么才可以精确得到相关的信息，这个要了解计算机图形学中的三次贝塞尔曲线的相关知识，类似与 photoshop 中的曲线调色。这里我们忽略。

还有一种不是平滑过渡，是跳跃式过渡，属性值为：steps(n,type)。第一个值是一个数值，表示跳跃几次。第二个值是 start 或者 end，可选值。表示开始时跳跃，还是结束时跳跃。

//跳跃 10 次至结束

```
transition-timing-function: steps(10,end);
```

五．**transition-delay**

这个属性可以设置一个过渡延迟效果，就是效果在设置的延迟时间后再执行。使用 transition-delay 属性值。如果有多个样式效果，可以设置多个延迟时间，以空格隔开。

//设置延迟效果

```
transition-delay: 0s, 1s, 0s;
```

六．简写和版本

我可以直接使用 transition 来简写，有两种形式的简写。第一种是，每个样式单独声明；第二种是不去考虑样式，即使用 all 全部声明。

//单独声明

```
transition: background-color 1s ease 0s, color 1s ease 0s, margin-left 1s ease 0s;
```

//如果每个样式都是统一的，直接使用 all

```
transition: all 1s ease 0s;
```

为了兼容旧版本，需要加上相应的浏览器前缀，版本信息如下表：

	Opera	Firefox	Chrome	Safari	IE
支持需带前缀	15 ~ 22	5 ~ 15	4 ~ 25	3.1 ~ 6	无
支持不带前缀	23+	16+	26+	6.1+	10.0+

//兼容完整版

```
-webkit-transition: all 1s ease 0s;
-moz-transition: all 1s ease 0s;
-o-transition: all 1s ease 0s;
-ms-transition: all 1s ease 0s;
transition: all 1s ease 0s;
```

第 26 章 CSS3 动画效果

学习要点：

1.动画简介

2.属性详解

3.简写和版本

主讲教师：李炎恢

本章主要探讨 HTML5 中 CSS3 的动画效果，可以通过类似 Flash 那样的关键帧模式控制运行。

一．动画简介

CSS3 提供了类似 Flash 关键帧控制的动画效果，通过 animation 属性实现。那么之前的 transition 属性只能通过指定属性的初始状态和结束状态来实现动画效果，有一定的局限性。

animation 实现动画效果主要由两个部分组成：1.通过类似 Flash 动画中的关键帧声明一个动画；2.在 animation 属性中调用关键帧声明的动画。

CSS3 提供的 animation 是一个复合属性，它包含了很多子属性。如下表所示：

属性	说明
animation-name	用来指定一个关键帧动画的名称，这个动画名必须对应一个@keyframes 规则。CSS 加载时会应用 animation-name 指定的动画，从而执行动画。
animation-duration	用来设置动画播放所需的时间
animation-timing-function	用来设置动画的播放方式
animation-delay	用来指定动画的延迟时间
animation-iteration-count	用来指定动画播放的循环次数
animation-direction	用来指定动画的播放方向
animation-play-state	用来控制动画的播放状态
animation-fill-mode	用来设置动画的时间外属性
animation	以上的简写形式

除了这 9 个属性之外，动画效果还有一个重要的属性，就是关键帧属性：@keyframes。它的作用是声明一个动画，然后在 animation 调用关键帧声明的动画。

//基本格式，“name”可自定义

```
@keyframes name {
/*...*/
}
```

二．属性详解

在讲解动画属性之前，先创建一个基本的样式。

//一个 div 元素

```
<div>我是 HTML5</div>
```

//设置 CSS

```
div { width: 200px; height: 200px; border: 1px solid green;
}
```

1.@keyframes

//创建动画的第一步，先声明一个动画关键帧。

```
@keyframes myani { 0% {
        margin-left:0px;
    } 50% { margin-left:100px;
    } 100% { margin-left:0px;
    } }
```

//或者重复的，可以并列写在一起

```
@keyframes myani { 0%, 100% {
        margin-left:0px;
    } 50% { background-color: black; margin-left:100px;
    } }
```

2.animation-name

//调用@keyframes 动画

```
animation-name: myani;
```

属性值	说明
none	默认值，没有指定任何动画
INDEX	是由@keyframes 指定创建的动画名称

3.animation-duration

//设置动画播放的时间

```
animation-duration: 1s;
```

当然，以上通过关键帧的方式，这里插入了三个关键帧。0%设置了白色和左偏移为 0，和初始状态一致，表明从这个地方开始动画。50%设置了黑色，左偏移 100px。而 100%则是最后一个设置，又回到了白色和左偏移为 0。整个动画就一目了然了。

而对于关键帧的用法，大部分用百分比比较容易控制，当然，还有其他一些控制方法。

//从什么状态过渡到什么状态

```
@keyframes myani { from {
        margin-left:0px;
    } to { margin-left:100px;
    } }
```

4.animation-timing-function

//设置缓动

```
animation-timing-function: ease-in;
```

属性值	说明
ease	默认值，元素样式从初始状态过渡到终止状态时速度由快到慢，逐渐变慢。等同于贝塞尔曲线(0.25, 0.1, 0.25, 1.0)
linear	元素样式从初始状态过渡到终止状态速度是恒速。等同于贝塞尔曲线(0.0, 0.0, 1.0, 1.0)
ease-in	元素样式从初始状态过渡到终止状态时，速度越来越快，呈一种加速状态。等同于贝塞尔曲线(0.42, 0, 1.0, 1.0)
ease-out	元素样式从初始状态过渡到终止状态时，速度越来越慢，呈一种减速状态。等同于贝塞尔曲线(0, 0, 0.58, 1.0)
ease-in-out	元素样式从初始状态过渡到终止状态时，先加速，再减速。等同于贝塞尔曲线(0.42, 0, 0.58, 1.0)
cubic-bezier	自定义三次贝塞尔曲线

5.animation-delay

//设置延迟时间

```
animation-delay: 1s;
```

6.animation-iteration-count

//设置循环次数

```
animation-iteration-count: infinite;
```

属性值	说明
次数	默认值为 1
infinite	表示无限次循环

7.animation-direction

//设置缓动方向交替

```
animation-direction: alternate;
```

属性值	说明
normal	默认值，每次播放向前
alternate	一次向前，一次向后，一次向前，一次向后这样交替

8.animation-play-state

//设置停止播放动画

```
animation-play-state: paused;
```

9.animation-fill-mode

//设置结束后不在返回

```
animation-fill-mode: forwards;
```

属性值	说明
none	默认值，表示按预期进行和结束
forwards	动画结束后继续应用最后关键帧位置，即不返回
backforwards	动画结束后迅速应用起始关键帧位置，即返回
both	根据情况产生 forwards 或 backforwards 的效果

//both 需要结合，次数和播放方向 animation-iteration-count: 4; animation-direction: alternate;

六．简写和版本

//简写形式完整版

```
animation: myani 1s ease 2 alternate 0s both;
```

为了兼容旧版本，需要加上相应的浏览器前缀，版本信息如下表：

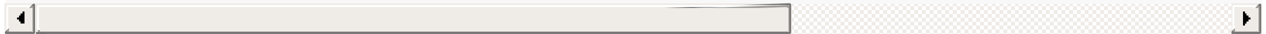
	Opera	Firefox	Chrome	Safari	IE
支持需带前缀	15 ~ 29	5 ~ 15	4 ~ 42	4 ~ 8	无
支持不带前缀	无	16+	43+	无	10.0+

//兼容完整版，Opera 在这个属性上加入 webkit，所以没有

```
-o--webkit-animation: myani 1s ease 2 alternate 0s both;
-moz-animation: myani 1s ease 2 alternate 0s both;
-ms-animation: myani 1s ease 2 alternate 0s both;
transition: all 1s ease 0s;
```

//@keyframes 也需要加上前缀

```
@-webkit-keyframes myani {...} @-moz-keyframes myani {...} @-o-keyframes myani {...} @-ms
```



第 27 章 CSS 传统布局[上]

学习要点：

1.布局模型

2.表格布局

3.浮动布局

主讲教师：李炎恢

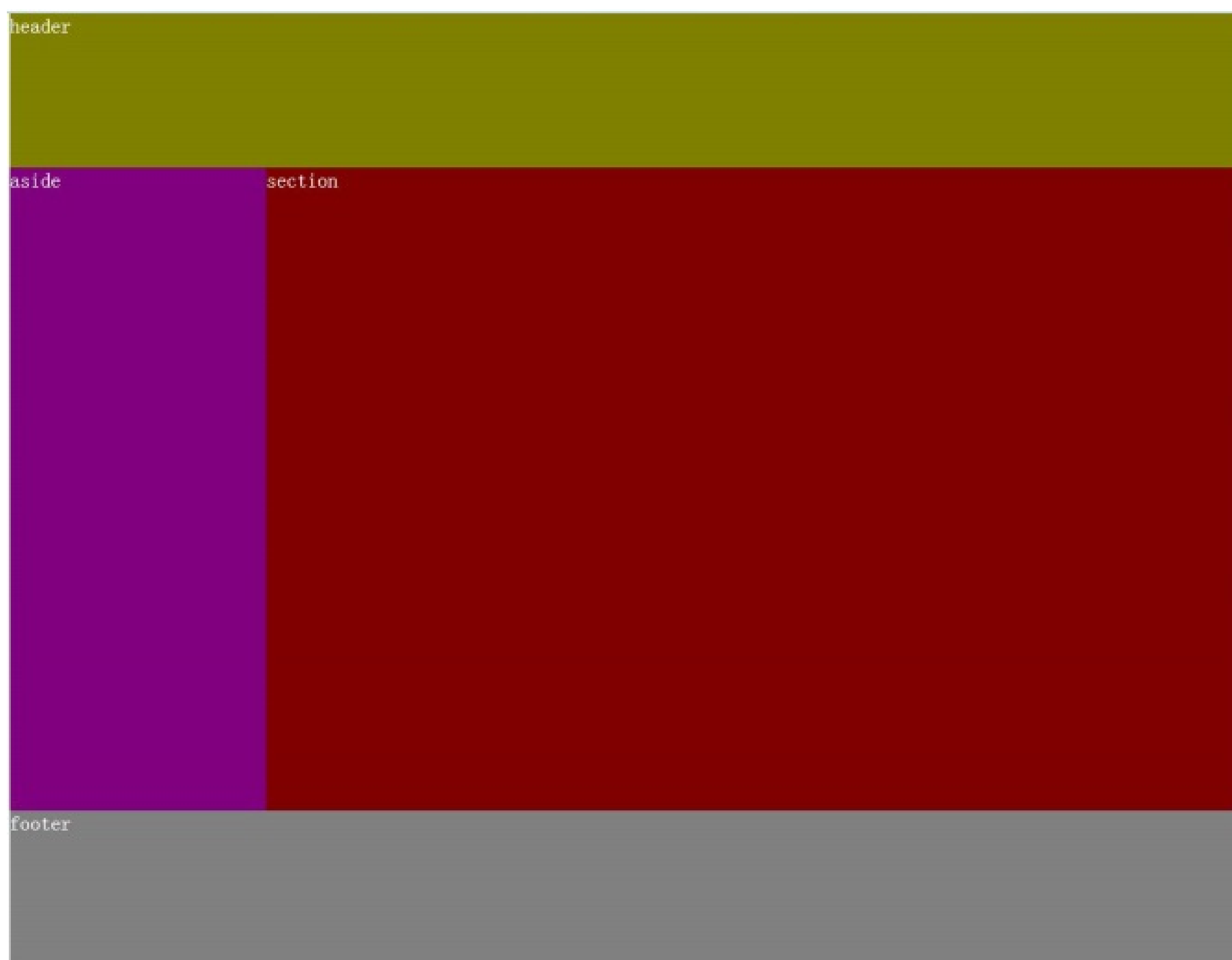
本章主要探讨 HTML5 中 CSS 早期所使用的传统布局，很多情况下，这些布局方式还是非常有用的。

一．布局模型

在早期没有平板和智能手机等移动设备大行其道的时期，Web 页面的设计主要是面向 PC 端电脑分辨率展开的。这种分辨率比例比较单一，基本上只要满足最低分辨率设计即可。一般来说有 4:3、16:10、16:9 这样的主要分辨率。那么，从这种比例上来看，长度总是大于宽度的。从最低分辨率 1024 * 768 设计即可。为了使浏览器底部不出现滚动条，需要减去适当的宽度，比如减去 28，最终固定长度设置为 996 即可。当然，也有一些网站在近两年讲最低分辨率设置为 1280 减去滚动条宽度，因为大显示器逐步主流。

除了刚才所说的固定长度的布局，还有一种是流体布局，就是布局的长度为百分比，比如 100%。不管你是什么分辨率，它都能全屏显示，当然，局限性也特别大，只适合一些单一页面，复杂的页面，会随着不同浏览器产生各种阅读障碍。

我们创建一个三行两列模型。并采用表格布局和浮动布局，构建固定和流体布局的方式，模型图如下：



二．表格布局

表格布局，就是通过设定固定的单元格，去除表格边框和填充实现的布局。当然这个布局非常不建议使用，只是教学了解。表格应该用它最为语义的地方，就是二维表的数据显示。

1.固定布局

//HTML 部分

```
<table border="0">
    <tr>
        <td colspan="2" class="header">header</td>
    </tr>
    <tr>
        <td class="aside">aside</td>
        <td class="section">section</td>
    </tr>
    <tr>
        <td colspan="2" class="footer">footer</td>
    </tr>
</table>
```

//CSS 部分

```
body { margin: 0;
} table { margin: 0 auto; width: 960px; border-spacing: 0;
} .header { height: 120px;
} .aside { width: 200px; height: 500px;
} .section { width: 760px; height: 500px;
} .footer { height: 120px;
}
```

2.流体布局

表格的固定布局改成流体布局非常简单，只需要设置 table 为 100%即可。

//修改 table

```
table { width: 100%;
}
```

三·浮动布局

浮动布局主要采用 float 和 clear 两个属性来构建。

1.固定布局

//HTML 部分

```
<header> header </header>

<aside> aside </aside>

<section> section </section>

<footer> footer </footer>
```

//CSS 部分

```
body { width: 960px; margin: 0 auto; color: white;
} header { height: 120px;
} aside { width: 200px; height: 500px; float: left;
} section { width: 760px; height: 500px; float: right;
} footer { height: 120px; clear: both;
}
```

2.**流体布局**

流体布局只要更改 body 元素的限定长度为 auto 或 100%。然后左右两列分别设置 20% 和 80%即可。

//CSS 部分

```
body { width: auto;
} aside { width: 20%;
} section { width: 80%;
}
```

第 27 章 CSS 传统布局[下]

学习要点：

1.定位布局

2.box-sizing

3.resize

主讲教师：李炎恢

本章主要探讨 HTML5 中 CSS 早期所使用的传统布局，很多情况下，这些布局方式还是非常有用的。

一．定位布局

在使用定位布局前，我们先了解一下定位属性的用法。CSS2 提供了 position 属性来实现元素的绝对定位和相对定位。

属性	说明
static	默认值，无定位。
absolute	绝对定位，使用 top、right、bottom、left进行位移。
relative	相对定位，使用 top、right、bottom、left进行位移。
fixed	以窗口参考定位，使用 top、right、bottom、left 进行位移。

//绝对定位，脱离文档流，以窗口文档左上角 0,0 为起点

```
header { position: absolute; top: 100px; left: 100px;
}
```

所谓脱离文档流的意思，就是本身这个元素在文档流是占位的。如果脱离了，就不占有文档的位置，好像浮在了空中一般，有了层次感。

由于绝对定位脱离了文档流，出现层次概念。那么每个元素到底在那一层，会不会冲突覆盖。这时通过 z-index 属性来判定它们的层次关系。

属性	说明
auto	默认层次
数字	设置层次，数字越大，层次越高

//设置在 100 层上

```
header { z-index: 100;
}
```

//以窗口参考定位，脱离文档流，会随着滚动条滚动而滚动

```
header { position: fixed; top: 100px; left: 100px;
}
```

//相对定位，不脱离文档流，占位偏移

```
header { position: relative; top: 100px; left: 100px;
}
```

这三种分别都在各自的情况下使用，均比较常用。但还有一种情况，就是：1.既要脱离文档流（这样元素之间不会相互冲突）；2.以父元素，比如 body 或其他父元素为参考点（这样可以实现区域性绝对定位）；3.还必须是绝对定位。

//第一步，将需要设置参考点的父元素设置为相对，且不设置坐标

```
body { position: relative;
}
```

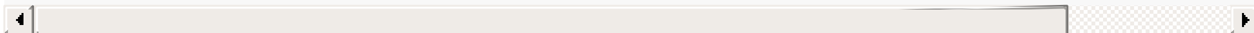
//第二步，如果父元素设置了参考点，子元素的绝对定位将以它为基准

```
header { position: absolute; top: 0px; left: 0px;
}
```

1.**固定布局**

//CSS 部分

```
body { width: 960px; margin: 0 auto; position: relative;
} header { width: 960px; height: 120px; position: absolute; top: 0; left: 0;
} aside { width: 200px; height: 500px; position: absolute; top: 120px; left: 0;
} section { width: 760px; height: 500px; position: absolute; top: 120px; /*left: 200px;*/
} footer { width: 960px; height: 120px; position: absolute; top: 620px;
}
```



在上面，基本都用了定位来进行固定布局。但细心的可以发现，其实只有右侧需要实行绝对定位，其他就按照普通的摆放即可。对于设计成流体布局，只要将长度设置成百分比即可。

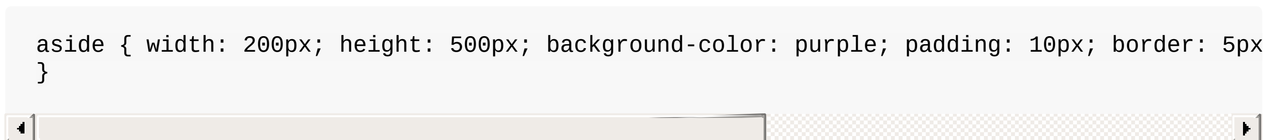
二．**box-sizing**

在盒模型那个章节，我们了解到元素盒子如果加入了内边距 padding 和边框 border 后，它的总长度会增加。那么如果这个元素用于非常精确的布局时，我们就需要进行计算增减。这其实是比较烦人的操作，尤其是动态设置页面布局的时候。

CSS3 提供了一个属性 box-sizing，这个属性可以定义元素盒子的解析方式，从而可以选择避免掉布局元素盒子增加内边距和边框的长度增减问题。

属性	说明
content-box	默认值，border 和 padding 设置后用于元素的总长度。
border-box	border 和 padding 设置后不用于元素的总长度。

//设置 border-box 让 border 和 padding 不在额外增加元素大小

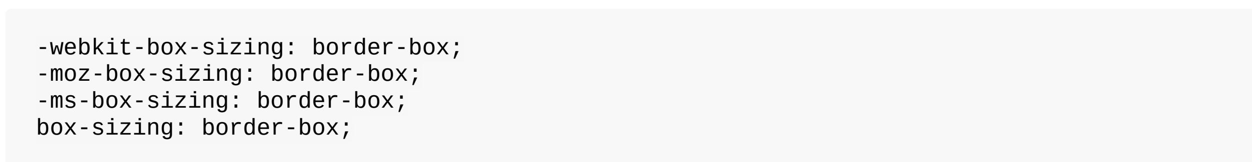


```
aside { width: 200px; height: 500px; background-color: purple; padding: 10px; border: 5px
}
```

box-sizing 是 CSS3 推出的，各个厂商在实现时设置了私有前缀。

	Opera	Firefox	Chrome	Safari	IE
支持需带前缀	无	2 ~ 28	4 ~ 9	3.1 ~ 5	8.0+
支持不带前缀	10.1+	29+	10+	6+	9.0+

//完整形式



```
-webkit-box-sizing: border-box;
-moz-box-sizing: border-box;
-ms-box-sizing: border-box;
box-sizing: border-box;
```

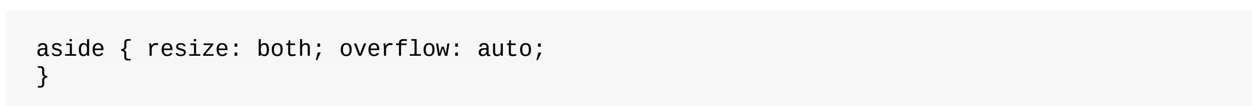
三．**resize**

CSS3 提供了一个 resize 属性，来更改元素尺寸大小。

属性	说明
none	默认值，不允许用户调整元素大小。
both	用户可以调节元素的宽度和高度。
horizontal	用户可以调节元素的宽度。
vertical	用户可以调节元素的高度。

一般普通元素，默认值是不允许的。但如果是表单类的 textarea 元素，默认是允许的。而普通元素需要设置 overflow:auto，配合 resize 才会出现可拖拽的图形。

//允许修改



```
aside { resize: both; overflow: auto;
}
```

第 28 章 CSS3 多列布局

学习要点：

1.早期多列问题

2.属性及版本

3.属性解释

主讲教师：李炎恢

本章主要探讨 HTML5 中 CSS3 提供的多列布局，通过多列布局我们方便的创建流体的多列布局。

一．早期多列问题

我们有时想布局成报纸、杂志那样的多列方式（至少两列，一般三列以上），但早期 CSS 提供的布局方式都有着极大的限制。如果是固体布局，那么使用浮动或定位布局都可以完成。但对于流体的多列，比如三列以上，那只能使用浮动布局进行，而它又有极大的限制。因为它是区块性的，对于动态的内容无法伸缩自适应，基本无能力。

//五段内容，分为三列

```
<div>
    <p> 我是第一段内容....省略的部分复制大量文本 </p>
    <p> 我是第二段内容....省略的部分复制大量文本 </p>
    <p> 我是第三段内容....省略的部分复制大量文本 </p>
    <p> 我是第四段内容....省略的部分复制大量文本 </p>
    <p> 我是第五段内容....省略的部分复制大量文本 </p>
</div>
```

//带标题的五段内容，分为三列

```
<div>
    <h4>第一段标题</h4>
    <p> 我是第一段内容....省略的部分复制大量文本 </p>
    <h4>第二段标题</h4>
    <p> 我是第二段内容....省略的部分复制大量文本 </p>
    <h4>第三段标题</h4>
    <p> 我是第三段内容....省略的部分复制大量文本 </p>
    <h4>第四段标题</h4>
    <p> 我是第四段内容....省略的部分复制大量文本 </p>
    <h4>第五段标题</h4>
    <p> 我是第五段内容....省略的部分复制大量文本 </p>
</div>
```

上门两组内容，如果想用浮动和定位实现流体三列，基本不可能。并且实际应用中，需求可能多变，要更改成四列或者五列呢？所以，CSS3 提供了多列布局属性 columns 来实现这个动态变换的功能。

二・属性及版本

CSS3 提供了 columns 多列布局属性等 7 个属性集合，具体如下：

属性	说明
columns	集成 column-width 和 column-count 两个属性
column-width	定义每列列宽度
column-count	定义分分列列数
column-gap	定义列间距
column-rule	定义列边框
column-span	定义多列布局中子元素跨列效果，firefox 不支持
column-fill	控制每列的列高效果，主流浏览器不支持

由于 column 属性集尚未纳入标准，大多数浏览器必须使用厂商前缀才能访问，好在主流的浏览器都可以很好的支持了。下面是主流浏览器的支持和前缀情况。

	Opera	Firefox	Chrome	Safari	IE
支持需带前缀	11.5 ~ 29	2 ~ 40	4 ~ 45	3.1 ~ 8	无
支持不带前缀	无	无	无	无	10.0+

通过上面的表格，我们可以了解到，要想让最新的浏览器全部实现效果，都必须使用前缀。

//完整形式

```
-webkit-columns: 150px 4;
-moz-columns: 150px 4;
columns: border-box;
```

三・属性解释

为了方便演示，我们在 Firefox 火狐浏览器测试，只用-moz-前缀演示。

1.columns

columns 是一个复合属性，包含 columns-width 和 columns-count 这两种简写。意为同时设置分列列数和分列宽度。//分成四列，每列宽度自适应

```
-moz-columns: auto 4;
```

2.column-width

column-width 属性，用于设置每列的宽度。

//设置列宽

```
-moz-column-width: 200px;
```

这里设置了 200px，有点最小宽度的意思。当浏览器缩放到小于 200 大小时，将变成 1 列显示。而如果是 auto，则一直保持四列。

属性	说明
auto	默认值，自适应。
长度值	设置列宽。

3.column-count

column-count 属性，用于设置多少列。

//设置列数

```
-moz-column-count: 4;
```

属性值	说明
auto	默认值，表示就 1 列。
数值	设置列数。

4.column-gap

column-gap 属性，用于设置列间距

//设置列间距

```
-moz-column-gap: 100px;
```

属性值	说明
auto	默认值，表示就 1 列。
数值	设置列数。

5.column-rule

column-rule 属性，设置每列中间的分割线//设置列边线

```
-moz-column-rule: 2px dashed gray;
```

属性	说明
column-rule	<宽度> <样式> <颜色>的边框简写形式。
column-rule-width	单独设置边框宽度。
column-rule-style	单独设置边框的样式。
column-rule-color	单独设置边框的颜色。

列边线不会影响到布局，它会根据布局的缩放自我调整是否显示。如果我们把页面缩放到只能显示一列时，它自动消失了。

6.column-span

column-span 属性，设置跨列大标题。

//跨列标题，由于火狐尚未支持，固使用 webkit

```
-webkit-column-span: all;
```

属性	说明
none	默认值，表示不跨列。
all	表示跨列。

第 29 章 CSS3 弹性伸缩布局[上]

学习要点：

1.布局简介

2.旧版本

主讲教师：李炎恢

本章主要探讨 HTML5 中 CSS3 提供的用来实现未来响应式弹性伸缩布局方案，这里做一个初步的了解。

一．布局简介

CSS3 提供一种崭新的布局方式：Flexbox 布局，即弹性伸缩布局模型(Flexible Box)。用来提供一个更加有效的方式实现响应式布局。但是用于这个布局方式还处于 W3C 的草案阶段，并且它还分为旧版本、新版本以及混合过渡版本三种不同的编码方式。在发展中，可能还有各种改动，浏览器的兼容性还存在问题。所以，本节课作为初步了解即可。

首先，我们来看下旧版本的浏览器兼容情况：

Flexbox 旧版本兼容情况

属性	IE	Firefox	Chrome	Opera	Safari
带前缀	无	4 ~ 25	4 ~ 31	15 ~ 18	5.17+
不带前缀	无	无	无	无	无

以上的数据，我们摘自 CSS3 手册上的。当然，不同的教材和文章的会略有不同。但误差率也就一到两个版本，影响不大。

首先，第一步：先创建一段内容，分成三个区域。

//HTML 部分

```
<div>
    <p> 第一段内容... </p>
    <p> 第二段内容... </p>
    <p> 第三段内容... </p>
</div>
```

//CSS 部分

```
p { width: 150px; border: 1px solid gray; margin: 5px; padding: 5px;
} div { display: -moz-box; display: -webkit-box; display: box;
}
```

通过以上设置，在除了 IE 浏览器外，布局实现了水平分布。那么下面，我们就重点研究一下这些属性的含义。

二．旧版本

如果属性和属性值为：display:box，那么就是 2009 年 7 月份设定的工作草案，属于旧版本。它面向的是一些早期浏览器的弹性布局方案。

首先，我们要将几个需要布局模块的父元素设置一下容器属性 display。

属性值	说明
box	将容器盒模型作为块级弹性伸缩盒显示（旧版本）
inline-box	将容器盒模型作为内联级弹性伸缩盒显示（旧版本）

我们知道块级它是占用整行的，类似<div>元素；而内联级不占用整行，类似<span>元素。但是我们设置了整个盒子，他们都不占用，保持一致。

//设置弹性，以火狐为例

```
div { display: -moz-box;
}
```

1.box-orient 属性

box-orient 主要实现盒子内部元素的流动方向。

//设置垂直流动

```
div { -webkit-box-orient: vertical;
}
```

属性值	说明
horizontal	伸缩项目从左到右水平排列
vertical	伸缩项目从上到下垂直排列
inline-axis	伸缩项目沿着内联轴排列显示
block-axis	伸缩项目沿着块轴排练显示

2.box-direction

box-direction 属性主要是设置伸缩容器中的流动顺序。

//设置逆序

```
div { -moz-box-direction: reverse;
}
```

属性值	说明
normal	默认值，正常顺序
reverse	逆序

3.box-pack

box-pack 属性用于伸缩项目的分布方式。

//分布方式已结束位置靠齐

```
div { -moz-box-pack: end;
}
```

属性值	说明
start	伸缩项目以起始点靠齐
end	伸缩项目以结束点靠齐
center	伸缩项目以中心点靠齐
justify	伸缩项目平局分布，-webkit-支持，-moz-不支持

4.box-align

box-align 属性用来处理伸缩容器的额外空间。//居中对齐，清理上下额外空间

```
div { -moz-box-align: center;
}
```

属性值	说明
start	伸缩项目以顶部为基准，清理下部额外空间
end	伸缩项目以底部为基准，清理上部额外空间
center	伸缩项目以中部为基准，平均清理上下部额外空间
baseline	伸缩项目以基线为基准，清理额外的空间
stretch	伸缩项目填充整个容器，默认

5.box-flex

box-flex 属性可以使用浮点数分配伸缩项目的比例//设置每个伸缩项目占用的比例

```
p:nth-child(1) { -moz-box-flex: 1;
} p:nth-child(2) { -moz-box-flex: 2.5;
} p:nth-child(3) { -moz-box-flex: 1;
}
```

6.box-ordinal-group

box-ordinal-group 属性可以设置伸缩项目的显示位置。

//将第一个位置的元素，跳转到第三个位置

```
p:nth-child(1) { -moz-box-ordinal-group: 3;
}
```

第 29 章 CSS3 弹性伸缩布局[中]

学习要点：

1.混合过度版

主讲教师：李炎恢

本章主要探讨 HTML5 中 CSS3 提供的用来实现未来响应式弹性伸缩布局方案，这里做一个初步的了解。

一．混合过渡版

混合版本的 Flexbox 模型是 2011 年提出的工作草案，主要是针对 IE10 浏览器实现的伸缩布局效果，其功能和旧版本的属性大同小异。我们还是采用上一节课的文件，然后使用混合过渡代码，实现 IE10 的伸缩布局。

首先，设置伸缩盒的 display 有如下两个属性值：

属性值	说明
flexbox	将容器盒模型作为块级弹性伸缩盒显示（混合版本）
inline-flexbox	将容器盒模型作为内联级弹性伸缩盒显示（混合版本）

//需要 IE 前缀-ms-

```
div { display: -ms-flexbox;
}
```

1.flex-direction

flex-direction 属性和旧版本 box-orient 属性一样，都是设置伸缩项目的排列方式。

//设置从上往下排列

```
div { -ms-flex-direction: column;
}
```

属性值	说明
row	设置从左到右排列
row-reverse	设置从右到左排列
column	设置从上到下排列
column-reverse	设置从下到上排列

2.flex-wrap

flex-wrap 属性类似与旧版本中的 box-lines，但是 box-lines 我们没有讲解，原因是没有浏览器支持它。

//设置无法容纳时，自动换行

```
div { -ms-flex-wrap: wrap;
}
```

属性值	说明
nowrap	默认值，都在一行或一列显示
wrap	伸缩项目无法容纳时，自动换行
wrap-reverse	伸缩项目无法容纳时，自动换行，方向和 wrap 相反

3.flex-flow

flex-flow 属性是集合了排列方向和控制换行的简写形式。

//简写形式

```
div { -ms-flex-flow: row wrap;
}
```

4.flex-pack

flex-pack 属性和旧版本中的 box-pack 一样，设置伸缩项目的对其方式。

//按照中心点对齐

```
div { -ms-flex-pack: center;
}
```

属性值	说明
start	伸缩项目以起始点靠齐
end	伸缩项目以结束点靠齐
center	伸缩项目以中心点靠齐
justify	伸缩项目平局分布

5.flex-align

flex-align 属性和旧版本中的 box-align 一样，处理伸缩项目容器的额外空间。

//处理额外空间

```
div { -ms-flex-align: center;
}
```

属性值	说明
start	伸缩项目以顶部为基准，清理下部额外空间
end	伸缩项目以底部为基准，清理上部额外空间
center	伸缩项目以中部为基准，平均清理上下部额外空间
baseline	伸缩项目以基线为基准，清理额外的空间
stretch	伸缩项目填充整个容器，默认

6.flex

flex 属性和旧版本中的 box-flex 类似，用来控制伸缩容器的比例分配。

//设置比例分配

```
p:nth-child(1) { -ms-flex: 1;
} p:nth-child(2) { -ms-flex: 3;
} p:nth-child(3) { -ms-flex: 1;
}
```

7.flex-order

flex-order 属性和 box-ordinal-group 属性一样控制伸缩项目出现的顺序。

//设置伸缩项目顺序

```
p:nth-child(1) { -ms-flex-order: 2;
} p:nth-child(2) { -ms-flex-order: 3;
} p:nth-child(3) { -ms-flex-order: 1;
}
```

第 29 章 CSS3 弹性伸缩布局[下]

学习要点：

1.新版本

主讲教师：李炎恢

本章主要探讨 HTML5 中 CSS3 提供的用来实现未来响应式弹性伸缩布局方案，这里做一个初步的了解。

一．新版本

新版本的 Flexbox 模型是 2012 年 9 月提出的工作草案，这个草案是由 W3C 推出的最新语法。这个版本立志于指定标准，让新式的浏览器全面兼容，在未来浏览器的更新换代中实现了统一。

首先，设置伸缩盒的 display 有如下两个属性值：

属性值	说明
flex	将容器盒模型作为块级弹性伸缩盒显示（新版本）
inline-flex	将容器盒模型作为内联级弹性伸缩盒显示（新版本）

//大部分不需要前缀

```
div { display: flex;
}
```

属性	IE	Firefox	Chrome	Opera	Safari
带前缀	无	无	21 ~ 28	无	7.0
不带前缀	11+	20+	29+	12.1	无

从这张表可以看出，只有 webkit 引擎的浏览器 Chrome 和 Safari 的版本需要添加-webkit-，而 Chrome 浏览器 29 版本以后可以省略了。

1.flex-direction

flex-direction 属性和旧版本 box-orient 属性一样，都是设置伸缩项目的排列方式。

//设置从上往下排列

```
div { flex-direction: column;
}
```

属性值	说明
row	设置从左到右排列
row-reverse	设置从右到左排列
column	设置从上到下排列
column-reverse	设置从下到上排列

2.flex-wrap

flex-wrap 属性类似与旧版本中的 box-lines，但是 box-lines 我们没有讲解，原因是没有浏览器支持它。

//设置无法容纳时，自动换行

```
div { -ms-flex-wrap: wrap;
}
```

属性值	说明
nowrap	默认值，都在一行或一列显示
wrap	伸缩项目无法容纳时，自动换行
wrap-reverse	伸缩项目无法容纳时，自动换行，方向和 wrap 相反

3.flex-flow

flex-flow 属性是集合了排列方向和控制换行的简写形式。

//简写形式

```
div { flex-flow: row wrap;
}
```

4.justify-content

justify-content 属性和旧版本中的 box-pack 一样，设置伸缩项目的对其方式。

//按照中心点对齐

```
div { justify-content: space-around;
}
```

属性值	说明
flex-start	伸缩项目以起始点靠齐
flex-end	伸缩项目以结束点靠齐
center	伸缩项目以中心点靠齐
space-between	伸缩项目平局分布
space-around	同上，但两端保留一半的空间

5.align-items

align-items 属性和旧版本中的 box-align 一样，处理伸缩项目容器的额外空间。

//处理额外空间

```
div { align-itmes: center;
}
```

属性值	说明
flex-start	伸缩项目以顶部为基准，清理下部额外空间
flex-end	伸缩项目以底部为基准，清理上部额外空间
center	伸缩项目以中部为基准，平均清理上下部额外空间
baseline	伸缩项目以基线为基准，清理额外的空间
stretch	伸缩项目填充整个容器，默认

6.align-self

align-self 和 align-items 一样，都是清理额外空间，但它是单独设置某一个伸缩项目的。所有的值和 align-itmes 一致。

//单独设置清理额外空间

```
p:nth-child(2) { align-self: center;
}
```

7.flex

flex 属性和旧版本中的 box-flex 类似，用来控制伸缩容器的比例分配。

//设置比例分配

```
p:nth-child(1) { flex: 1;
} p:nth-child(2) { flex: 3;
} p:nth-child(3) { flex: 1;
}
```

8.order

order 属性和 box-ordinal-group 属性一样控制伸缩项目出现的顺序。

//设置伸缩项目顺序

```
p:nth-child(1) { order: 2;
} p:nth-child(2) { order: 3;
} p:nth-child(3) { order: 1;
}
```

第 30 章 使用 Emmet 插件

学习要点：

1.安装方式

2.自定义！生成

3.快速生成

主讲教师：李炎恢

本章主要探讨了解一下 Sublime Text3 的一个 HTML5 代码提示插件：Emmet，这个插件比自带原生的要强大许多。

一．安装方式

Emmet 插件安装一般采用两种方式，1.通过命令安装；2.下载离线安装。我这里直接采用的是下载离线安装方式。具体步骤：

1.解压下载好的 Emmet 插件包(这里会提供)；

2.将 Emmet 和 PyV8 两个文件夹复制到 Sublime Text3 的程序包中；

3.左下角会显示自动安装，安装好后，重启 Sublime Text3；4.在编辑器输入英文状态下的“!”，然后 ctrl+e，出现了 HTML5 的代码库，则安装

成功。

注：如果安装失败或出现其他错误，请自行百度选用其他方式安装，或解决安装出错的问题。

二．自定义！生成

我们输入!，然后 ctrl+e，默认情况下会出现如下代码：

//默认代码

```
<!doctype html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Document</title>
    </head>
    <body>

    </body>
</html>
```

这里有两个地方和我们之前生成的代码不一样，第一处是：doctype 没有大写；第二处：lang 是 en 的。其实这两处不改也没有太大关系，但有强迫症的看了可能会难受。具体修改方法如下：

1.首先，进入程序包 pagkages；

2.其次，进入 Emmet 文件夹，再进入 emmet 文件夹，找到 snippets.json 文件；

3.最后，打开这个文件，找到相应处修改即可。

三 . 快速生成

Emmet 提供了非常丰富的 HTML 和 CSS 代码的快速生成功能，通过使用快速生成代码，极大的增加了开发速度。只不过，Emmet 提供的生成方式需要二次学习，起初可能还不如手工敲击的快。所以，需要一定时间的学习磨合。

//快速生成 HTML5 代码结构

! + (ctrl + e 或 tab 键) 或 html:5 + tab 键

```
<!DOCTYPE html>
<html lang="zh-cn">
    <head>
        <meta charset="UTF-8">
        <title>Document</title>
    </head>
    <body>

    </body>
</html>
```

所有代码生成，都需要通过 tab 键来生成代码，后面不在赘述。

//快速生成标签代码 a

```
<a href=""></a>
```

//快速生成标签相应的属性值a:link、a:mail

```
<a href="http://"></a>

&lt;a href="mailto:"&gt;&lt;/a&gt;
```

//生成标签内的值a{超链接}

```
<a href="">超链接</a>
```

//生成 CSS 链接 link link

```
<link rel="stylesheet" href="">
```

//生成表单控件input、input:hidden

```
<input type="text">
<input type="hidden" name="">
```

//生成带子标签的一组标签ul+、ol+、dl+、table+

```
<ul>
    <li></li>
</ul>

<ol>
    <li></li>
</ol>

<dl>
    <dt></dt>
    <dd></dd>
</dl>

<table>
    <tr>
        <td></td>
    </tr>
</table>
```

//生成嵌套子标签 nav>ul>li

```
<nav>
    <ul>
        <li></li>
    </ul>
</nav>
```

//生成相邻兄弟标签 div+p+h1

```
<div></div>

<p></p>

<h1></h1>
```

//生成乘积数量的标签 ul>li*5

```
<ul>
    <li></li>

    <li></li>

    <li></li>

    <li></li>

    <li></li>
</ul>
```

//创建具有 ID 的标签 div#header

```
<div id="header"></div>
```

//创建具有 class 的标签 div.header、div.header.sidebar

```
<div class="header"></div>

<div class="header sidebar"></div>
```

以上是 HTML 部分的代码生成功能，下面来看下 CSS 的快速生成功能：

//生成 position: relative pos

输入 pos 即可出现 position:relative 这组 CSS 样式，并且 relative 是选定状态，有助于你更换属性值。

但是我们发现使用 sublime 结合 Emmet 插件的 CSS 提示非常的灵活，不会死板的必须要输入 pos。下面的输入都可以得到相应的值：

po = position: relative

只要输入 po 或者大于 po 字母量的值，都可以得到 position:relative。当然，如果你只是输入 p，那么由于优先级的考虑，出现的是 padding: |。

如果你输入有误，它也会自动纠错，比如下面这个：

pod = position: relative

如果你想一开始得到的是 absolute 这个属性值，那么直接输入：

poa = position: absolute;

当然，上门的标准写法是这样的：

pos:a = position: absolute;

如果想输入背景的属性，直接使用 bg 即可：

bg = background: |;

使用 bg+可以展开背景属性的完整形式： bg+ = background: #fff url() 0 0 no-repeat;

使用 bg:n 可以设置背景属性值为 none： bg:n = background: none;

备注：HTML 和 CSS 其他大部分代码生成方案，可以参考如下网址：

http://docs.emmet.io/cheat-sheet/

Bootstrap 教程

第 1 章 Bootstrap 介绍

学习要点：

1.Bootstrap 概述

2.Bootstrap 特点

3.Bootstrap 结构

4.创建第一个页面

5.学习的各项准备

主讲教师：李炎恢

本节课我们主要了解一下 Boostrap 历史、特点、用途，以及为什么选择 Boostrap 来开发我们的 Web 项目。

一．**Bootstrap 概述**

Bootstrap 是由 Twitter 公司(全球最大的微博)的两名技术工程师研发的一个基于HTML、CSS、JavaScript 的开源框架。该框架代码简洁、视觉优美，可用于快速、简单地构建基于 PC 及移动端设备的 Web 页面需求。

2010 年 6 月，Twitter 内部的工程师为了解决前端开发任务中的协作统一问题。经历各种方案后，Bootstrap 最终被确定下来，并于 2011 年 8 月发布。经过很长时间的迭代升级，由最初的 CSS 驱动项目发展成为内置很多 JavaScript 插件和图标的多功能 Web 前端的开源框架。

Bootstrap 最为重要的部分就是它的响应式布局，通过这种布局可以兼容 PC 端、PAD以及手机移动端的页面访问。

Bootstrap 下载及演示：

国内文档翻译官网：http://www.bootcss.com

瓢城 Web 俱乐部官网：http://www.ycku.com

二．**Bootstrap 特点**

Bootstrap 非常流行，得益于它非常实用的功能和特点。主要核心功能特点如下：

(1).跨设备、跨浏览器

可以兼容所有现代浏览器，包括比较诟病的 IE7、8。当然，本课程不再考虑 IE9 以下浏览器。

(2).响应式布局

不但可以支持 PC 端的各种分辨率的显示，还支持移动端 PAD、手机等屏幕的响应式切换显示。

(3).提供的全面的组件

Bootstrap 提供了实用性很强的组件，包括：导航、标签、工具条、按钮等一系列组件，方便开发者调用。

(4).内置 jQuery 插件

Bootstrap 提供了很多实用性的 jquery 插件，这些插件方便开发者实现 Web 中各种常规特效。

(5).支持 HTML5、CSS3

HTML5 语义化标签和 CSS3 属性，都得到很好的支持。

(6).支持 LESS 动态样式

LESS 使用变量、嵌套、操作混合编码，编写更快、更灵活的 CSS。它和 Bootstrap 能很好的配合开发。

三．**Bootstrap 结构**

首先，想要了解 Boostrap 的文档结构，需要在官网先把它下载到本地。Bootstrap下载地址如下：

Bootstrap 下载地址：http://v3.bootcss.com/ (选择生产环境即可，v3.3.4)

解压后，目录呈现这样的结构：

```
bootstrap/
```

```
├── css/
││├── bootstrap.css
││├── bootstrap.css.map
││├── bootstrap.min.css
││├── bootstrap-theme.css
││├── bootstrap-theme.css.map
││└── bootstrap-theme.min.css
├── js/
││├── bootstrap.js
││└── bootstrap.min.js
└── fonts/
    ├── glyphicons-halflings-regular.eot
    ├── glyphicons-halflings-regular.svg
    ├── glyphicons-halflings-regular.ttf
    ├── glyphicons-halflings-regular.woff
    └── glyphicons-halflings-regular.woff2
```

主要分为三大核心目录：**css(样式)、js(脚本)、fonts(字体)**。

1.css 目录中有四个 css 后缀的文件，其中包含 min 字样的，是压缩版本，一般使用这个；不包含的属于没有压缩的，可以学习了解 css 代码的文件；而 map 后缀的文件则是css 源码映射表，在一些特定的浏览器工具中使用。

2.js 目录包含两个文件，是未压缩和压缩的 js 文件。

3.fonts 目录包含了不同后缀的字体文件。

四．创建第一个页面

我们创建一个 HTML5 的页面，并且将 Bootstrap 的核心文件引入，然后测试是否正常显示。

//第一个 Bootstrap

```
<!DOCTYPE html>
<html lang="zh-cn">
<head>
    <meta charset="UTF-8">
    <title>Bootstrap 介绍</title>
    <link rel="stylesheet" href="css/bootstrap.min.css">
</head>
<body>
    <button class="btn btn-info">Bootstrap</button>
    <script src="js/jquery.min.js"></script>
    <script src="js/bootstrap.min.js"></script>
</body>
</html>
```

五 · 学习的各项准备

1.开发工具， 我们使用Sublime Text3作为Bootstrap的开发工具， 并且安装了Emmet代码生成插件；

2.测试工具，除了常规的现代浏览器，其次就是作为移动端的测试工具，我们这里采用的是 Opera Mobile Emulator，和 Chrome 的移动 Web 测试工具。

3.所需基础，至少有 HTML5 第一季课程的基础，Bootstrap 内置了很多 jQuery 插件，虽然使用起来比 jQuery 或 JS 本身要简单的多，但如果有 jQuery 和 JS 课程的基础，那学习理解力就更加深入；

4.课程分辨率：基础课程，我们使用 1024 x 768 来录制，但某些特殊部分，比如响应式和项目课程，需要大分辨率录制，会采用 1440 x 900 来录制。

第 2 章 排版样式

学习要点：

1.页面排版

主讲教师：李炎恢

本节课我们主要学习一下 Bootstrap 全局 CSS 样式中的排版样式，包括了标题、页面主体、对齐、列表等常规内容。

一．页面排版

Bootstrap 提供了一些常规设计好的页面排版的样式供开发者使用。

1.页面主体****

Bootstrap 将全局 font-size 设置为 14px，line-height 行高设置为 1.428(即20px)；<p>段落元素被设置等于 1/2 行高(即 10px)；颜色被设置为#333。

//创建包含段落突出的文本

```
<p>Bootstrap 框架</p>
<p class="lead">Bootstrap 框架</p>
<p>Bootstrap 框架</p>
<p>Bootstrap 框架</p>
<p>Bootstrap 框架</p>
```

2.标题****

//从 h1 到 h6

```
<h1>Bootstrap 框架</h1>//36px <h2>Bootstrap 框架</h2>//30px <h3>Bootstrap 框架</h3>//24px <
```



我们从 Firebug 查看元素了解到， Bootstrap 分别对 h1 ~ h6 进行了 CSS 样式的重构，并且还支持普通内联元素定义 class=(.h1 ~ h6)来实现相同的功能。

//内联元素使用标题字体

```
<span class="h1">Bootstrap</span>
```

注：通过 Firebug 查看元素还看到，字体颜色、字体样式、行高均被固定了，从而保证了统一性，而原生的会根据系统内置的首选字体决定，颜色是最黑色。

在 h1 ~ h6 元素之间，还可以嵌入一个 small 元素作为副标题，

//在标题元素内插入 small 元素

```
<h1>Bootstrap 框架 <small>Bootstrap 小标题</small></h1>
<h2>Bootstrap 框架 <small>Bootstrap 小标题</small></h2>
<h3>Bootstrap 框架 <small>Bootstrap 小标题</small></h3>
<h4>Bootstrap 框架 <small>Bootstrap 小标题</small></h4>
<h5>Bootstrap 框架 <small>Bootstrap 小标题</small></h5>
<h6>Bootstrap 框架 <small>Bootstrap 小标题</small></h6>
```

通过 Firebug 查看，我们发现 h1 ~ h3 下 small 元素的大小只占父元素的 65%，那么通过计算(查看 Firebug 计算后的样式)， h1 ~ h3 下的 small 为 23.4px、 19.5px、 15.6px；h4 ~ h6 下 small 元素的大小只占父元素的 75% ,分别为：13.5px、10.5px、9px。在 h1 ~ h6 下的 small 样式也进行了改变，颜色变成淡灰色：#777，行高为 1，粗度为 400。

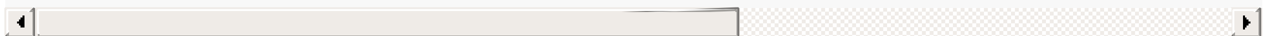
3.内联文本元素

//添加标记，<mark>元素或.mark 类

```
<p>Bootstrap<mark>框架</mark></p>
```

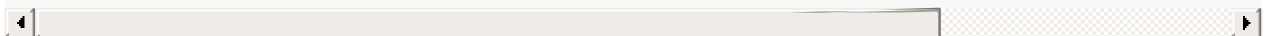
//各种加线条的文本

```
<del>Bootstrap 框架</del>//删除的文本 <s>Bootstrap 框架</s>//无用的文本 <ins>Bootstrap 框架</i
```



//各种强调的文本

```
<small>Bootstrap 框架</small>//标准字号的 85% <strong>Bootstrap 框架</strong>//加粗 700 <em>B
```



4.对齐

//设置文本对齐

```
<p class="text-left">Bootstrap 框架</p>//居左 <p class="text-center">Bootstrap 框架</p>//居
```



5.大小写

//设置英文文本大小写

```
<p class="text-lowercase">Bootstrap 框架</p> //小写 <p class="text-uppercase">Bootstrap 框
```



6.缩略语

//缩略语

```
Bootstrap<abbr title="Bootstrap" class="initialism">框架</abbr>
```

7.地址文本

//设置地址，去掉了倾斜，设置了行高，底部 20px

```
<address>
   <strong>Twitter, Inc.</strong><br>    795 Folsom Ave, Suite 600<br>    San Francisco,
   <abbr title="Phone">P:</abbr> (123) 456-7890 </address>
```

8.引用文本

//默认样式引用，增加了做边线，设定了字体大小和内外边距

```
<blockquote>    Bootstrap 框架 </blockquote>
```

//反向

```
<blockquote class="blockquote-reverse ">    Bootstrap 框架 </blockquote>
```

9.列表排版

//移出默认样式

```
<ul class="list-unstyled">
   <li>Bootstrap 框架</li>
   <li>Bootstrap 框架</li>
   <li>Bootstrap 框架</li>
   <li>Bootstrap 框架</li>
   <li>Bootstrap 框架</li>
</ul>
```

//设置成内联

```
<ul class="list-inline">
   <li>Bootstrap 框架</li>
   <li>Bootstrap 框架</li>
   <li>Bootstrap 框架</li>
   <li>Bootstrap 框架</li>
   <li>Bootstrap 框架</li>
</ul>
```

//水平排列描述列表

```
<dl class="dl-horizontal">
   <dt>Bootstrap</dt>
   <dd>Bootstrap 提供了一些常规设计好的页面排版的样式供开发者使用。</dd>
</dl>
```

10.代码

//内联代码

```
<code>&lt;section&gt;</code>
```

//用户输入

```
press <kbd>ctrl + ,</kbd>
```

//代码块

```
<pre>&lt;p&gt;Please input...&lt;/p&gt;</pre>
```

Bootstrap 还列举了<var>表示标记变量，<samp>表示程序输出，只不过没有重新复写 CSS。

第 3 章 表格和按钮

学习要点：

1.表格

2.按钮

主讲教师：李炎恢

本节课我们主要学习一下 Bootstrap 表格和按钮功能，通过内置的 CSS 定义，显示各种丰富的效果。

一．表格

Bootstrap 提供了一些丰富的表格样式供开发者使用。

1.基本格式

//实现基本的表格样式

```
<table class="table">
```

注：我们可以通过 Firebug 查看相应的 CSS。

2.条纹状表格

//让<tbody>里的行产生一行隔一行加单色背景效果

```
<table class="table table-striped">
```

注：表格效果需要基于基本格式.table

3.带边框的表格

//给表格增加边框

```
<table class="table table-bordered">
```

4.悬停鼠标

//让<tbody>下的表格悬停鼠标实现背景效果

```
<table class="table table-hover">
```

5.状态类

//可以单独设置每一行的背景样式

```
<tr class="success">
```

注：一共五种不同的样式可供选择。

样式说明:

active鼠标悬停在行或单元格上

success标识成功或积极的动作

info标识普通的提示信息或动作

warning标识警告或需要用户注意

danger表示危险或潜在的带来负面影响的动作

6.隐藏某一行

//隐藏行

```
<tr class="sr-only">
```

7.响应式表格

//表格父元素设置响应式，小于 768px 出现边框

```
<body class="table-responsive">
```

二 · 按钮

Bootstrap 提供了很多丰富按钮供开发者使用。

1.可作为按钮使用的标签或元素

//转化成普通按钮

```
<a href="###" class="btn btn-default">Link</a>
<button class="btn btn-default">Button</button>
<input type="button" class="btn btn-default" value="input">
```

注意事项有三点：

(1).针对组件的注意事项

虽然按钮类可以应用到 <a> 和 <button> 元素上，但是，导航和导航条组件只支持<button>元素。

(2).链接被作为按钮使用时的注意事项

如果 <a> 元素被作为按钮使用 -- 并用于在当前页面触发某些功能 -- 而不是用于链接其他页面或链接当前页面中的其他部分，那么，务必为其设置 role="button" 属性。

(3).跨浏览器展现

我们总结的最佳实践是：强烈建议尽可能使用 <button> 元素来获得在各个浏览器上获得相匹配的绘制效果。

另外，我们还发现了 Firefox <30 版本的浏览器上出现的一个 bug，其表现是：阻止我们为基于 <input> 元素所创建的按钮设置 line-height 属性，这就导致在

Firefox 浏览器上不能完全和其他按钮保持一致的高度。

2.预定义样式

//一般信息

```
<button class="btn btn-info">Button</button>
```

样式说明

btn-default默认样式

btn-success成功样式

btn-info一般信息样式

btn-warning警告样式

btn-danger危险样式

btn-primary首选项样式

btn-link链接样式

3.尺寸大小

//从大到小的尺寸

```
<button class="btn btn-lg">Button</button>
<button class="btn">Button</button>
<button class="btn btn-sm">Button</button>
<button class="btn btn-xs">Button</button>
```

4.块级按钮

//块级换行

```
<button class="btn btn-block">Button</button>
<button class="btn btn-block">Button</button>
```

5.激活状态

//激活按钮

```
<button class="btn active">Button</button>
```

6.禁用状态

//禁用按钮

```
<button class="btn active disabled">Button</button>
```

第 4 章 表单和图片

学习要点：

1.表单

2.图片

主讲教师：李炎恢

本节课我们主要学习一下 Bootstrap 表单和图片功能，通过内置的 CSS 定义，显示各种丰富的效果。

一．表单

Bootstrap 提供了一些丰富的表单样式供开发者使用。

1.基本格式

//实现基本的表单样式

```
<form>
    <div class="form-group">
        <label>电子邮件</label>
        <input type="email" class="form-control" placeholder="请输入您的电子邮件">
    </div>
    <div class="form-group">
        <label>密码</label>
        <input type="password" class="form-control" placeholder="请输入您的密码">
    </div>
</form>
```

注：只有正确设置了输入框的 type 类型，才能被赋予正确的样式。支持的输入框控件包括：text、password、datetime、datetime-local、date、month、time、week、number、email、url、search、tel 和 color。

2.内联表单

//让表单左对齐浮动，并表现为 inline-block 内联块结构

```
<form class="form-inline">
```

注：当小于 768px，会恢复独占样式

3.表单合组

//前后增加片段

```
<div class="input-group">
    <div class="input-group-addon">¥</div>
    <input type="text" class="form-control">
    <div class="input-group-addon">.00</div>
</div>
```

4.水平排列

//让表单内的元素保持水平排列

```
<form class="form-horizontal">
    <div class="form-group">
        <label class="col-sm-2 control-label">电子邮件</label>
        <div class="col-sm-10">
            <input type="email" class="form-control" placeholder="请输入您的电子邮件">
        </div>
    </div>
</form>
```

注：这里用到了 col-sm 栅格系统，后面章节会重点讲解，而 control-label 表示和父元素样式同步。

5.复选框和单选框

//设置复选框，在一行

```
<div class="checkbox">
    <label> <input type="checkbox">体育 </label>
</div>
<div class="checkbox">
    <label> <input type="checkbox">音乐 </label>
</div>
```

//设置禁用的复选框

```
<div class="checkbox disabled">
    <label> <input type="checkbox" disabled>音乐 </label>
</div>
```

//设置内联一行显示的复选框

```
<label class="checkbox-inline"> <input type="checkbox">体育</label>
<label class="checkbox-inline disabled"> <input type="checkbox" disabled>音乐</label>
```

//设置单选框

```
<div class="radio disabled">
    <label> <input type="radio" name="sex" disabled>男</label>
</div>
```

6.下拉列表

//设置下拉列表

```
<select class="form-control">
    <option>1</option>
    <option>2</option>
    <option>3</option>
    <option>4</option>
    <option>5</option>
</select>
```

7.校验状态

//设置为错误状态

```
<div class="form-group has-error">
```

注：还有其他状态如下:

样式说明

has-error错误状态

has-success成功状态

has-warning警告状态

//label 标签同步相应状态

```
<label class="control-label">Input with success</label>
```

8.添加额外的图标

//文本框右侧内置文本图标

```
<div class="form-group has-feedback">
    <label>电子邮件</label>
    <input type="email" class="form-control">
    <span class="glyphicon glyphicon-ok form-control-feedback"></span>
</div>
```

注：除了 glyphicon-ok 外，还有几个如下表：

样式说明

glyphicon-ok成功状态

glyphicon-warning-sign警告状态

glyphicon-remove错误状态

9.控制尺寸

//从大到小

```
<input type="password" class="form-control input-lg">
<input type="password" class="form-control">
<input type="password" class="form-control input-sm">
```

注：也可以设置父元素 form-group-lg、form-group-sm，来调整。

二．图片

Bootstrap 提供了一些丰富的图片样式供开发者使用。

1.图片形状

//三种形状

```
<img src="img/pic.png" alt="图片" class="img-rounded">
<img src="img/pic.png" alt="图片" class="img-circle">
<img src="img/pic.png" alt="图片" class="img-thumbnail">
```

//响应式图片

```
<img src="img/pic.png" alt="图片" class="img-responsive">
```

第 5 章 栅格系统

学习要点：

1.移动设备优先

2.布局容器

3.栅格系统

主讲教师：李炎恢

本节课我们主要学习一下 Bootstrap 的栅格系统，提供了一套响应式、移动设备优先的流式栅格系统。

一．移动设备优先

在 HTML5 的项目中，我们做了移动端的项目。它有一份非常重要的 meta，用于设置屏幕和设备等宽以及是否运行用户缩放，及缩放比例的问题。

//分别为：屏幕宽度和设备一致、初始缩放比例、最大缩放比例和禁止用户缩放

```
<meta name="viewport" content="width=device-width, initial-scale=1,maximum-scale=1, user-
```

二．布局容器

Bootstrap 需要为页面内容和栅格系统包裹一个.container 容器。由于 padding 等属性的原因，这两种容器类不能相互嵌套。

//固定宽度

```
<div class="container"> ... </div>
```

//100%宽度

```
<div class="container-fluid"> ... </div>
```

栅格系统中， 浏览器会随着屏幕的大小的增减自动分配最多12列。 通过一系列的行(row)与列(column)的组合来创建页面布局。工作原理如下：

1.“行 （row） ” 必须包含在 .container （固定宽度） 或 .container-fluid （100%宽度）中，以便为其赋予合适的排列（aligment）和内补（padding） 。

2.通过“行（row）”在水平方向创建一组“列（column）”。

3.你的内容应当放置于“列（column）”内，并且，只有“列（column）”可以作为行（row）”的直接子元素。

4.类似 .row 和 .col-xs-4 这种预定义的类，可以用来快速创建栅格布局。

Bootstrap 源码中定义的 mixin 也可以用来创建语义化的布局。

5.通过为“列 （column） ”设置 padding 属性， 从而创建列与列之间的间隔 （gutter） 。通过为 .row 元素设置负值 margin 从而抵消掉为 .container 元素设置的 padding，也就间接为“行（row）”所包含的“列（column）”抵消掉了 padding。

6.负值的 margin 就是下面的示例为什么是向外突出的原因。在栅格列中的内容排成一行。

7.栅格系统中的列是通过指定 1 到 12 的值来表示其跨越的范围。例如，三个等宽的列可以使用三个 .col-xs-4 来创建。

8.如果一“行（row）”中包含了的“列（column）”大于 12，多余的“列（column） ”所在的元素将被作为一个整体另起一行排列。

9.栅格类适用于与屏幕宽度大于或等于分界点大小的设备 ， 并且针对小屏幕设备覆盖栅格类。 因此，在元素上应用任何 .col-md- 栅格类适用于与屏幕宽度大于或等于分界点大小的设备 ，并且针对小屏幕设备覆盖栅格类。因此， 在元素上应用任何 *.col-lg-*不存在， 也影响大屏幕设备。

//创建一个响应式行

```
<div class="container">
    <div class="row"> ... </div>
</div>
```

//创建最多 12 列的响应式行

```
<div class="container">
    <div class="row">
          <div class="col-md-1 a">1</div>
          <div class="col-md-1 a">2</div>
          <div class="col-md-1 a">3</div>
          <div class="col-md-1 a">4</div>
          <div class="col-md-1 a">5</div>
          <div class="col-md-1 a">6</div>
          <div class="col-md-1 a">7</div>
          <div class="col-md-1 a">8</div>
          <div class="col-md-1 a">9</div>
          <div class="col-md-1 a">10</div>
          <div class="col-md-1 a">11</div>
          <div class="col-md-1 a">12</div>
    </div>
</div>
```

//为了显示明显的 CSS

```
.a { height: 100px; background-color: #eee; border: 1px solid #ccc;
}
```

//总列数都是 12，每列分配多列

```
<div class="container">
    <div class="row">
          <div class="col-md-4 a">1-4</div>
          <div class="col-md-4 a">5-8</div>
          <div class="col-md-4 a">9-12</div>
    </div>
    <div class="row">
          <div class="col-md-8 a">1-8</div>
          <div class="col-md-4 a">9-12</div>
    </div>
</div>
```

栅格参数表

如上图所示，栅格系统最外层区分了四种宽度的浏览器：超小屏(<768px)、小屏(>=768px)、中屏(>=992px)和大屏(>=1200px)。而内层.container 容器的自适应宽度为：自动、750px、970px 和 1170px。自动的意思为，如果你是手机屏幕，则全面独占一行显示。

//四种屏幕分类全部激活

```
<div class="container">
    <div class="row">
          <div class="col-lg-3 col-md-4 col-sm-6 col-xs-12 a">4</div>
          <div class="col-lg-3 col-md-4 col-sm-6 col-xs-12 a">4</div>
          <div class="col-lg-3 col-md-4 col-sm-6 col-xs-12 a">4</div>
          <div class="col-lg-3 col-md-4 col-sm-6 col-xs-12 a">4</div>
          <div class="col-lg-3 col-md-4 col-sm-6 col-xs-12 a">4</div>
          <div class="col-lg-3 col-md-4 col-sm-6 col-xs-12 a">4</div>
          <div class="col-lg-3 col-md-4 col-sm-6 col-xs-12 a">4</div>
          <div class="col-lg-3 col-md-4 col-sm-6 col-xs-12 a">4</div>
          <div class="col-lg-3 col-md-4 col-sm-6 col-xs-12 a">4</div>
          <div class="col-lg-3 col-md-4 col-sm-6 col-xs-12 a">4</div>
          <div class="col-lg-3 col-md-4 col-sm-6 col-xs-12 a">4</div>
          <div class="col-lg-3 col-md-4 col-sm-6 col-xs-12 a">4</div>
    </div>
</div>
```

//有时我们可以设置列偏移，让中间保持空隙

```
<div class="container">
    <div class="row">
          <div class="col-md-8 a">8</div>
          <div class="col-md-3 col-md-offset-1 a">3</div>
    </div>
</div>
```

//也可以嵌套，嵌满也是 12 列

```
<div class="container">
    <div class="row">
          <div class="col-md-9 a">
              <div class="col-md-8 a">1-8</div>
              <div class="col-md-4 a">9-12</div>
          </div>
          <div class="col-md-3 a"> 11-12 </div>
    </div>
</div>
```

//可以把两个列交换位置，push 向左移动，pull 向右移动

```
<div class="container">
    <div class="row">
          <div class="col-md-9 col-md-push-3 a">9</div>
          <div class="col-md-3 col-md-pull-9 a">3</div>
    </div>
</div>
```

第 6 章 辅组类和响应式工具

学习要点：

1.辅组类

2.响应式工具

主讲教师：李炎恢

本节课我们主要学习一下 Bootstrap 的辅组类和响应式工具， 辅助类提供了一组类来辅组页面设计，而响应式工具则利用媒体查询显示或隐藏某些内容。

一．辅助类

Bootstrap 在布局方面提供了一些细小的辅组样式，用于文字颜色以及背景色的设置、显示关闭图标等等。

1.情景文本颜色

```
样式列表
 样式名          描述
   text-muted    柔和灰
 text-primary    主要蓝
 text-success    成功绿
    text-info    信息蓝
 text-warning    警告黄
  text-danger    危险红
```

//各种色调的字体

```
<p class="text-muted">Bootstrap 视频教程</p>
<p class="text-primary">Bootstrap 视频教程</p>
<p class="text-success">Bootstrap 视频教程</p>
<p class="text-info">Bootstrap 视频教程</p>
<p class="text-warning">Bootstrap 视频教程</p>
<p class="text-danger">Bootstrap 视频教程</p>
```

2.情景背景色

```
样式列表
样式名          描述
bg-primary    主要蓝
bg-success    成功绿
   bg-info    信息蓝
bg-warning    警告黄
 bg-danger    危险红
```

//各种色调的背景

```
<p class="bg-primary">Bootstrap 视频教程</p>
<p class="bg-success">Bootstrap 视频教程</p>
<p class="bg-info">Bootstrap 视频教程</p>
<p class="bg-warning">Bootstrap 视频教程</p>
<p class="bg-danger">Bootstrap 视频教程</p>
```

3.关闭按钮

```
<button type="button" class="close">&times;</button>
```

4.三角符号

```
<span class="caret"></span>
```

5.快速浮动

```
<div class="pull-left">左边</div>
<div class="pull-right">右边</div>
```

注：这个浮动其实就是 float，只不过使用了!important 加强了优先级。

6.块级居中

```
<div class="center-block">居中</div>
```

注：就是 margin:x auto；并且设置了 display:block;。

7.清理浮动

```
<div class="clearfix"></div>
```

注：这个 div 可以放在需要清理浮动区块的前面即可。

8.显示和隐藏

```
<div class="show">show</div>
<div class="hidden">hidden</div>
```

二．响应式工具

在媒体查询时，针对不同的屏幕大小，有时需要显示和隐藏部分内容。响应式工具类，就提供了这种解决方案。

	超小屏幕 手机 (<768px)	小屏幕 平板 (≥768px)	中等屏幕 桌面 (≥992px)	大屏幕 桌面 (≥1200px)
.visible-xs-*	可见	隐藏	隐藏	隐藏
.visible-sm-*	隐藏	可见	隐藏	隐藏
.visible-md-*	隐藏	隐藏	可见	隐藏
.visible-lg-*	隐藏	隐藏	隐藏	可见
.hidden-xs	隐藏	可见	可见	可见
.hidden-sm	可见	隐藏	可见	可见
.hidden-md	可见	可见	隐藏	可见
.hidden-lg	可见	可见	可见	隐藏

//超小屏幕激活显示

```
<div class="visible-xs-block a">Bootstrap</div>
```

//超小屏幕激活隐藏

```
<div class="hidden-xs a">Bootstrap</div>
```

注：对于显示的内容，有三种变体，分别为：block、inline-block、inline。

第 7 章 图标菜单按钮组件

学习要点：

1.小图标组件

2.下拉菜单组件

3.按钮组组件

4.按钮式下拉菜单

主讲教师：李炎恢

本节课我们主要学习一下 Bootstrap 的三个组件功能：小图标组件、下拉菜单组件和各种按钮组件。

一．小图标组件

Bootstrap 提供了免费的 263 个小图标（数了两次），具体可以参考中文官网的组件链接：http://v3.bootcss.com/components/#glyphicons。

部分图标如下：

glyphicon glyphicon-heart	glyphicon glyphicon-star	glyphicon glyphicon-star-empty	glyphicon glyphicon-user
glyphicon glyphicon-ok	glyphicon glyphicon-remove	glyphicon glyphicon-zoom-in	glyphicon glyphicon-zoom-out
glyphicon glyphicon-home	glyphicon glyphicon-file	glyphicon glyphicon-time	glyphicon glyphicon-road

可以使用<i>或<span>标签来配合使用，具体如下：

//使用小图标

```
<i class="glyphicon glyphicon-star"></i>
<span class="glyphicon glyphicon-star"></span>
```

//也可以结合按钮

```
<button class="btn btn-default btn-lg">
    <span class="glyphicon glyphicon-star"></span>
</button>
<button class="btn btn-default btn">
    <span class="glyphicon glyphicon-star"></span>
</button>
<button class="btn btn-default btn-sm">
    <span class="glyphicon glyphicon-star"></span>
</button>
<button class="btn btn-default btn-xs">
    <span class="glyphicon glyphicon-star"></span>
</button>
```

二．下拉菜单组件

下拉菜单，就是点击一个元素或按钮，触发隐藏的列表显示出来。

//基本格式

```
<div class="dropdown">
    <button class="btn btn-default" data-toggle="dropdown"> 下拉菜单 <span class="caret"></
    </button>
    <ul class="dropdown-menu">
        <li>
            <a href="#">首页</a>
        </li>
        <li>
            <a href="#">资讯</a>
        </li>
        <li>
            <a href="#">产品</a>
        </li>
        <li>
            <a href="#">关于</a>
        </li>
    </ul>
</div>
```

按钮和菜单需要包裹在.dropdown 的容器里，而作为被点击的元素按钮需要设置data-toggle="dropdown"才能有效。对于菜单部分，设置 class="dropdown-menu"才能自动隐藏并添加固定样式。设置 class="caret"表示箭头，可上可下。

//设置向上触发

```
<div class="dropup">
```

//菜单项居右对齐，默认值是 dropdown-menu-left

```
<ul class="dropdown-menu dropdown-menu-right">
```

//设置菜单的标题，不要加超链接

```
<li class="dropdown-header">网站导航</li>
```

//设置菜单的分割线

```
<li class="divider"></li>
```

//设置菜单的禁用项

```
<li class="disabled"><a href="#">产品</a></li>
```

//让菜单默认显示

```
<div class="dropdown open">
```

三．按钮组组件

按钮组就是多个按钮集成在一个容器里形成独有的效果。

//基本格式

```
<div class="btn-group">
    <button type="button" class="btn btn-default"> 左 </button>
    <button type="button" class="btn btn-default"> 中 </button>
    <button type="button" class="btn btn-default"> 右 </button>
</div>
```

//将多个按钮组整合起来便于管理

```
<div class="btn-toolbar">
    <div class="btn-group">
        <button type="button" class="btn btn-default"> 左 </button>
        <button type="button" class="btn btn-default"> 中 </button>
        <button type="button" class="btn btn-default"> 右 </button>
    </div>

    <div class="btn-group">
        <button type="button" class="btn btn-default"> 1 </button>
        <button type="button" class="btn btn-default"> 2 </button>
        <button type="button" class="btn btn-default"> 3 </button>
    </div>
</div>
```

//设置按钮组大小

```
<div class="btn-group btn-group-lg">
<div class="btn-group>
<div class="btn-group btn-group-sm">
<div class="btn-group btn-group-xs">
```

//嵌套一个分组，比如下拉菜单

```
<div class="btn-group">
    <button type="button" class="btn btn-default"> 左 </button>
    <button type="button" class="btn btn-default"> 中 </button>
    <button type="button" class="btn btn-default"> 右 </button>
    <div class="btn-group">
        <button class="btn btn-default dropdown-toggle" data-toggle="dropdown"> 下拉菜单 <
        </button>

        <ul class="dropdown-menu">
            <li>
                <a href="#">首页</a>
            </li>
            <li>
                <a href="#">资讯</a>
            </li>
            <li>
                <a href="#">产品</a>
            </li>
            <li>
                <a href="#">关于</a>
            </li>
        </ul>
    </div>
</div>
```

注意：这里<div>中并没有实现 class="dropdown"，通过源码分析知道嵌套本身已经有定位就不需要再设置。而右边的圆角只要多加一个 class="dropdown-toggle"即可。

//设置按钮组垂直排列

```
<div class="btn-group-vertical">
```

//设置两端对齐按钮组，使用<a>标签

```
<div class="btn-group-justified">
    <a type="button" class="btn btn-default">左</a>
    <a type="button" class="btn btn-default">中</a>
    <a type="button" class="btn btn-default">右</a>
</div>
```

//如果需要使用<button>标签，则需要对每个按钮进行群组

```
<div class="btn-group-justified">
    <div class="btn-group">
        <button type="button" class="btn btn-default"> 左 </button>
    </div>
    <div class="btn-group">
        <button type="button" class="btn btn-default"> 中 </button>
    </div>
    <div class="btn-group">
        <button type="button" class="btn btn-default"> 右 </button>
    </div>
</div>
```

四．按钮式下拉菜单

这个下拉菜单其实和第二个知识点一样，只不过，这个是在群组里，不需要<div>声明 class="dropdown"。

//群组按钮下拉菜单

```
<div class="btn-group">
    <button type="button" class="btn btn-default dropdown-toggle"data-toggle="dropdown">
    </button>
    <ul class="dropdown-menu">
        <li>
            <a href="#">首页</a>
        </li>
        <li>
            <a href="#">资讯</a>
        </li>
        <li>
            <a href="#">产品</a>
        </li>
        <li>
            <a href="#">关于</a>
        </li>
    </ul>
</div>
```

//分裂式按钮下拉菜单

```
<div class="btn-group">
    <button type="button" class="btn btn-default"> 下拉菜单 </button>
    <button type="button" class="btn btn-default dropdown-toggle"data-toggle="dropdown">
        <span class="caret"></span>
    </button>
    <ul class="dropdown-menu">
        <li>
            <a href="#">首页</a>
        </li>
        <li>
            <a href="#">资讯</a>
        </li>
        <li>
            <a href="#">产品</a>
        </li>
        <li>
            <a href="#">关于</a>
        </li>
    </ul>
</div>
```

//向上弹出式

```
<div class="btn-group dropup">
```

第 8 章 输入框和导航组件

学习要点：

1.输入框组件

2.导航组件

3.导航条组件

主讲教师：李炎恢

本节课我们主要学习一下Bootstrap的两个个组件功能： 输入框组件和导航导航条组件。

一．输入框组件

文本输入框就是可以在<input>元素前后加上文字或按钮，可以实现对表单控件的扩展。

//在左侧添加文字

```
<div class="input-group">
    <span class="input-group-addon">@</span>
    <input type="text" class="form-control">
</div>
```

//在右侧添加文字

```
<div class="input-group">
    <input type="text" class="form-control">
    <span class="input-group-addon">@163.com</span>
</div>
```

//在两侧添加文字

```
<div class="input-group">
    <span class="input-group-addon">$</span>
    <input type="text" class="form-control">
    <span class="input-group-addon">.00</span>
</div>
```

//设置尺寸，另外三种分别是默认、xs、sm

```
<div class="input-group input-group-lg">
```

//左侧使用复选框和单选框

```
<div class="input-group">
    <span class="input-group-addon">
        <input type="checkbox">
    </span>
    <input type="text" class="form-control">
</div>

<div class="input-group">
    <span class="input-group-addon">
        <input type="radio">
    </span>
    <input type="text" class="form-control">
</div>
```

//左侧使用按钮

```
<div class="input-group">
    <span class="input-group-btn">
        <button type="button" class="btn btn-default"> 按钮 </button> </span>
    <input type="text" class="form-control">
</div>
```

//左侧使用下拉菜单或分列式

```
<div class="input-group">
    <span class="input-group-btn">
        <button class="btn btn-default dropdown-toggle" data-toggle="dropdown"> 下拉菜单 <s
        </button>
        <ul class="dropdown-menu">
            <li class="dropdown-header"> 网站导航 </li>
            <li>
                <a href="#">首页</a>
            </li>
            <li>
                <a href="#">资讯</a>
            </li>
            <li class="divider">
                <a href="#">产品</a>
            </li>
            <li class="disabled">
                <a href="#">关于</a>
            </li>
        </ul> </span>
    <input type="text" class="form-control">
</div>
```

二．导航组件

Bootstrap 提供了一组导航组件，用于实现 Web 页面的栏目操作。

//基本导航标签页

```
<ul class="nav nav-tabs">
    <li class="active">
        <a href="#">首页</a>
    </li>
    <li>
        <a href="#">资讯</a>
    </li>
    <li>
        <a href="#">产品</a>
    </li>
    <li>
        <a href="#">关于</a>
    </li>
</ul>
```

//胶囊式导航

```
<ul class="nav nav-pills">
```

//垂直胶囊式导航

```
<ul class="nav nav-pills nav-stacked">
```

//导航两端对齐

```
<ul class="nav nav-tabs nav-justified">
```

//禁用导航中的项目

```
<li class="disabled"><a href="#">关于</a></li>
```

//带下拉菜单的导航

```
<ul class="nav nav-tabs">
    <li class="active">
        <a href="#">首页</a>
    </li>
    <li>
        <a href="#">资讯</a>
    </li>
    <li class="dropdown">
        <a href="#" class="dropdown-toggle" data-toggle="dropdown"> 下拉菜单 <span class="
        <ul class="dropdown-menu">
            <li>
                <a href="#">菜单一</a>
            </li>
            <li>
                <a href="#">菜单二</a>
            </li>
        </ul>
    </li>
</ul>
```

三．导航条组件

导航条是网站中作为导航页头的响应式基础组件。

//基本格式

```
<nav class="navbar navbar-default"> ... </nav>
```

//反色调导航

```
<nav class="navbar navbar-inverse"> ... </nav>
```

//基本导航条，包含标题和列表

```
<nav class="navbar navbar-default">
    <div class="container">
        <div class="navbar-header">
            <a href="#" class="navbar-brand">标题</a>
        </div>
        <ul class="nav navbar-nav">
            <li class="active">
                <a href="#">首页</a>
            </li>
            <li>
                <a href="#">资讯</a>
            </li>
            <li class="disabled">
                <a href="#">产品</a>
            </li>
            <li>
                <a href="#">关于</a>
            </li>
        </ul>
    </div>
</nav>
```

//导航条中使用表单

```
<form action="" class="navbar-form navbar-left">
    <div class="input-group">
        <input type="text" class="form-control">
        <span class="input-group-btn">
            <button type="submit" class="btn btn-default"> 提交 </button>
        </span>
    </div>
</form>
```

//导航中使用按钮

```
<button class="btn btn-default navbar-btn">按钮</button>
```

//导航中使用对齐方式，left 和 right

```
<button class="btn btn-default navbar-btn navbar-right">按钮</button>
```

//导航中使用一段文本

```
<p class="navbar-text">我是一段文本</p>
```

//非导航链接，一般需要置入文本区域内

```
<a href="#" class="navbar-link">非导航链接</a>
```

//将导航固定在顶部，下面的内容会自动上移

```
<nav class="navbar navbar-default navbar-fixed-top">
```

//将导航补丁在底部

```
<nav class="navbar navbar-default navbar-fixed-bottom">
```

//静态导航，和页面等宽的导航条，去掉了圆角

```
<nav class="navbar navbar-default navbar-static-top">
```

第 9 章 路径分页标签和徽章组件

学习要点：

1.路径组件

2.分页组件

3.标签组件

4.徽章组件

主讲教师：李炎恢

本节课我们主要学习一下 Bootstrap 的四个组件功能：路径组件、分页组件、标签组件和徽章组件。

一．路径组件

路径组件也叫做面包屑导航。

//面包屑导航

```
<ol class="breadcrumb">
    <li>
        <a href="#">首页</a>
    </li>
    <li>
        <a href="#">产品列表</a>
    </li>
    <li class="active"> 韩版 2015 年羊绒毛衣 </li>
</ol>
```

二．分页组件

分页组件可以提供带有展示页面的功能。

//默认分页

```
<ul class="pagination">
    <li>
        <a href="#">&laquo;</a>
    </li>
    <li>
        <a href="#">1</a>
    </li>
    <li>
        <a href="#">2</a>
    </li>
    <li>
        <a href="#">3</a>
    </li>
    <li>
        <a href="#">4</a>
    </li>
    <li>
        <a href="#">5</a>
    </li>
    <li>
        <a href="#">&raquo;</a>
    </li>
</ul>
```

//首选项和禁用

```
<li class="active"><a href="#">1</a></li>
<li class="disabled"><a href="#">2</a></li>
```

//设置尺寸，四种 lg、默认、sm 和 xs

```
<ul class="pagination pagination-lg">
```

//翻页效果

```
<ul class="pager">
    <li>
        <a href="#">上一页</a>
    </li>
    <li>
        <a href="#">下一页</a>
    </li>
</ul>
```

//对齐翻页链接

```
<ul class="pager">
    <li class="previous">
        <a href="#">上一页</a>
    </li>
    <li class="next">
        <a href="#">下一页</a>
    </li>
</ul>
```

//翻页项禁用

```
<li class="previous disabled"><a href="#">上一页</a></li>
```

三·标签

//在文本后面带上标签

```
<h3>标签 <span class="label label-default">new</span></h3>
```

//不同色调的标签

```
<h3>标签 <span class="label label-primary">new</span></h3>
<h3>标签 <span class="label label-success">new</span></h3>
<h3>标签 <span class="label label-info">new</span></h3>
<h3>标签 <span class="label label-warning">new</span></h3>
<h3>标签 <span class="label label-danger">new</span></h3>
```

四·徽章

//未读信息数量徽章

```
<a href="#">信息 <span class="badge">10</span></a>
```

//按钮中使用徽章

```
<button class="btn btn-success"> 提交 <span class="badge">3</span>
</button>
```

//激活状态自动适配色调

```
<ul class="nav nav-pills">
    <li class="active">
        <a href="#">首页 <span class="badge">2</span></a>
    </li>
    <li>
        <a href="#">资讯</a>
    </li>
</ul>
```

第 10 章 巨幕页头缩略图和警告框组件

学习要点：

1.巨幕组件

2.页头组件

3.缩略图组件

4.警告框组件

主讲教师：李炎恢

本节课我们主要学习一下 Bootstrap 的四个组件功能：巨幕组件、页头组件、缩略图组件和警告框组件。

一．巨幕组件

巨幕组件主要是展示网站的关键性区域。

//在固定的范围内，有圆角

```
<div class="container">
    <div class="jumbotron">
        <h2>网站标题</h2>
        <p> 这是一个学习性的网站！ </p>
        <p>
            <a href="#" class="btn btn-default">更多内容</a>
        </p>
    </div>
</div>
```

//100%全屏，没有圆角

```
<div class="jumbotron">
    <div class="container">
        <h2>网站标题</h2>
        <p> 这是一个学习性的网站！ </p>
        <p>
            <a href="#" class="btn btn-default">更多内容</a>
        </p>
    </div>
</div>
```

二．页头组件

//增加一些空间

```
<div class="page-header">
    <h1>大标题 <small>小标题</small></h1>
</div>
```

三·缩略图组件

//缩略图配合响应式

```
<div class="container">
    <div class="row">
        <div class="col-xs-6 col-md-3 col-sm-4">
            <div class="thumbnail">
                <img src="img/pic.png" alt="">
            </div>
        </div>
        <div class="col-xs-6 col-md-3 col-sm-4">
            <div class="thumbnail">
                <img src="img/pic.png" alt="">
            </div>
        </div>
        <div class="col-xs-6 col-md-3 col-sm-4">
            <div class="thumbnail">
                <img src="img/pic.png" alt="">
            </div>
        </div>
        <div class="col-xs-6 col-md-3 col-sm-4">
            <div class="thumbnail">
                <img src="img/pic.png" alt="">
            </div>
        </div>
    </div>
</div>
```

//自定义内容

```
<div class="container">
    <div class="row">
        <div class="col-xs-6 col-md-3 col-sm-4">
            <div class="thumbnail">
                <img src="img/pic.png" alt="">
                <div class="caption">
                    <h3>图文并茂</h3>
                    <p> 这是一个图片结合文字的缩略图 </p>
                    <p>
                        <a href="#" class="btn btn-default">进入</a>
                    </p>
                </div>
            </div>
        </div>
        <div class="col-xs-6 col-md-3 col-sm-4">
            <div class="thumbnail">
                <img src="img/pic.png" alt="">
                <div class="caption">
                    <h3>图文并茂</h3>
                    <p> 这是一个图片结合文字的缩略图 </p>
                    <p>
                        <a href="#" class="btn btn-default">进入</a>
                    </p>

                </div>
            </div>
        </div>
        <div class="col-xs-6 col-md-3 col-sm-4">
            <div class="thumbnail">
                <img src="img/pic.png" alt="">
                <div class="caption">
                    <h3>图文并茂</h3>
                    <p> 这是一个图片结合文字的缩略图 </p>
                    <p>
                        <a href="#" class="btn btn-default">进入</a>
                    </p>
                </div>
            </div>
        </div>
        <div class="col-xs-6 col-md-3 col-sm-4">
            <div class="thumbnail">
                <img src="img/pic.png" alt="">
                <div class="caption">
                    <h3>图文并茂</h3>
                    <p> 这是一个图片结合文字的缩略图 </p>
                    <p>
                        <a href="#" class="btn btn-default">进入</a>
                    </p>
                </div>
            </div>
        </div>
    </div>
</div>
```

四．警告框组件

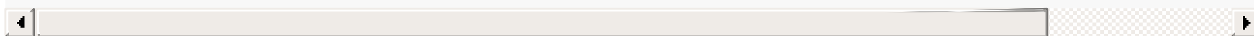
警告框组件是一组预定义消息。

//基本警告框

```
<div class="alert alert-success">Bootstrap</div>
<div class="alert alert-info">Bootstrap</div>
<div class="alert alert-warning">Bootstrap</div>
<div class="alert alert-danger">Bootstrap</div>
```

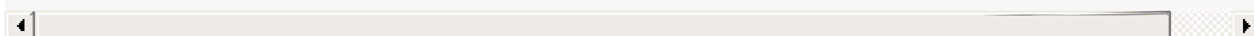
//带关闭的警告框

```
<div class="alert alert-success"> Bootstrap <button type="button" class="close" data-dism
        <span>&times;</span>
    </button>
</div>
```



//自动适配的超链接

```
<div class="alert alert-success"> Bootstrap，请到官网 <a href="#" class="alert-link">下载</
</div>
```



第 11 章 进度条媒体对象和 Well 组件

学习要点：

1.Well 组件

2.进度条组件

3.媒体对象组件

主讲教师：李炎恢

本节课我们主要学习一下 Bootstrap 的三个组件功能：Well 组件、进度条组件、媒体对象组件。

一．Well 组件

这个组件可以实现简单的嵌入效果。

//嵌入效果

```
<div class="well"> Bootstrap </div>
```

//有 lg 和 sm 两种可选值

```
<div class="well well-lg"> Bootstrap </div>
```

二．进度条组件

进度条组件为当前工作流程或动作提供时时反馈。

//基本进度条

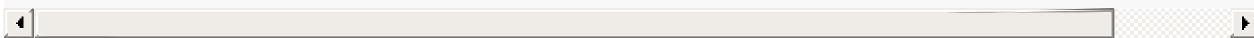
```
<div class="progress">
    <div class="progress-bar" style="width: 60%;"> 60% </div>
</div>
```

//最低值进度条

```
<div class="progress">
    <div class="progress-bar" style="min-width:20px"> 0% </div>
</div>
```

//结合情景的进度条

```
<div class="progress">
    <div class="progress-bar progress-bar-success" style="min-width:20px;width:60%"> 60%
</div>
```

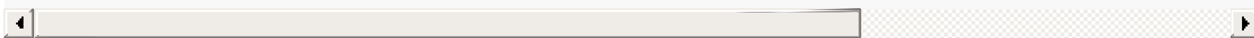


//条纹状，IE10+支持

```
<div class="progress">
    <div class="progress-bar progress-bar-success
    progress-bar-striped" style="min-width:20px;width:60%"> 60% </div>
</div>
```

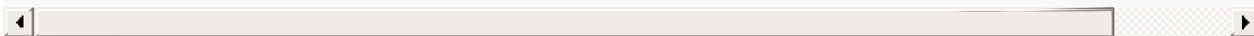
//动画效果

```
<div class="progress">
    <div class="progress-bar progress-bar-success progress-bar-stripedactive" style="min-
</div>
```



//堆叠效果

```
<div class="progress">
    <div class="progress-bar progress-bar-success" style="min-width:20px;width:35%"> 35%
    <div class="progress-bar progress-bar-warning" style="min-width:20px;width:20%"> 20%
    <div class="progress-bar progress-bar-danger" style="min-width:20px;width:10%"> 10% <
</div>
```



三．媒体对象组件

媒体对象可以包含图片、视频或音频等媒体，以达到对象和文本组合显示的样式效果。

//基本实例

```
<div class="media">
    <div class="media-left">
        <img src="img/small.png" alt="" class="media-object">
    </div>
    <div class="media-body">
        <h4 class="media-heading">标题</h4>
        <p> 企鹅（学名：Spheniscidae）：有“海洋之舟”美称的企鹅是一种最古老的游禽，它们很可能在地球穿上
    </div>
</div>
```



//媒体对象在右边

```
<div class="media">
    <div class="media-body">
        <h4 class="media-heading">标题</h4>

        <p> 企鹅（学名：Spheniscidae）：有“海洋之舟”美称的企鹅是一种最古老的游禽，它们很可能在地球穿上
    </div>
    <div class="media-right">
        <img src="img/small.png" alt="" class="media-object">
    </div>
</div>
```

//媒体对象列表

```
<ul class="media-list">
    <li class="media"> //将每个 media 存放在 media-body 内后即可
        ...代码较多，具体看视频 </li>
</ul>
```

第 12 章 列表组面板和嵌入组件

学习要点：

1.列表组组件

2.面板组件

3.响应式嵌入组件

主讲教师：李炎恢

本节课我们主要学习一下 Bootstrap 的三个组件功能：列表组组件、面板组件、 响应式嵌入组件。

一．列表组组件

列表组组件用于显示一组列表的组件。

//基本实例

```
<ul class="list-group">
    <li class="list-group-item"> 1.这是起始 </li>
    <li class="list-group-item"> 2.这是第二条数据 </li>
    <li class="list-group-item"> 3.这是第三排信息 </li>
    <li class="list-group-item"> 4.这是末尾 </li>
</ul>
```

//列表项带勋章

```
<li class="list-group-item"> 1.这是起始 <span class="badge">10</span>
</li>
```

//链接和首选

```
<div class="list-group">
    <a href="#" class="list-group-item active">1.这是起始 <span class="badge">10</span></a
    <a href="#" class="list-group-item">2.这是第二条数据</a>
    <a href="#" class="list-group-item">3.这是第三排信息</a>
    <a href="#" class="list-group-item">4.这是末尾</a>
</div>
```

//按钮式列表

```
<div class="list-group">
    <button class="list-group-item active"> 1.这是起始 <span class="badge">10</span>
    </button>
    <button class="list-group-item"> 2.这是第二条数据 </button>
    <button class="list-group-item"> 3.这是第三排信息 </button>
    <button class="list-group-item"> 4.这是末尾 </button>

</div>
```

//设置项目被禁用

```
class="list-group-item disabled"
```

//情景类

```
<li class="list-group-item list-group-item-success"> 3.这是第三排信息 </li>
```

//定制内容

```
<div class="list-group">
    <a href="#" class="list-group-item active"> <h4>内容标题</h4>
    <p class="list-group-item-text"> 这里是相关内容详情！ </p> </a>
    <a href="#" class="list-group-item"> <h4>内容标题</h4>
    <p class="list-group-item-text"> 这里是相关内容详情！ </p> </a>
    <a href="#" class="list-group-item"> <h4>内容标题</h4>
    <p class="list-group-item-text"> 这里是相关内容详情！ </p> </a>
</div>
```

二．面板组件

面板组件就是一个存放内容的容器组件。

//基本实例

```
<div class="panel panel-default">
    <div class="panel-body"> 这里是详细内容区！ </div>
</div>
```

//带标题容器的面板

```
<div class="panel panel-default">
    <div class="panel-heading"> 面板标题 </div>
    <div class="panel-body"> 这里是详细内容区！ </div>
</div>
```

//也可以设置标题元素

```
<div class="panel-heading">
    <h3 class="panel-title">面板标题</h3>
</div>
```

//带注脚的面板

```
<div class="panel-footer"> 这里是底部 </div>
```

//情景效果：default、success、info、warning、danger、primary

```
<div class="panel panel-success">
```

//表格类面板

```
<div class="panel panel-default">
    <div class="panel-heading"> 表格标题 </div>
    <div class="panel-body">
        <p> 这里是表格标题的详细内容！ </p>
    </div>
    <table class="table">
        <tr>
            <th>1</th>
            <th>2</th>
            <th>3</th>
        </tr>
        <tr>
            <td>1</td>
            <td>2</td>
            <td>3</td>
        </tr>
    </table>
</div>
```

//列表类面板

```
<div class="panel panel-default">
    <div class="panel-heading"> 表格标题 </div>
    <div class="panel-body">
        <p> 这里是表格标题的详细内容！ </p>
    </div>
    <ul class="list-group">

        <li class="list-group-item"> 1.这里是首页 </li>
        <li class="list-group-item"> 2.这里是第二个项目 </li>
        <li class="list-group-item"> 3.这里是第三个项目 </li>
        <li class="list-group-item"> 4.这里是第四个项目 </li>
    </ul>
</div>
```

三．响应式嵌入组件

根据被嵌入内容的外部容器的宽度， 自动创建一个固定的比例， 从而让浏览器自动确定内容的尺寸，能够在各种设备上缩放。

这些规则可以直接用于<iframe>、<embed>、<video>和<object>元素。

//16:9 响应式

```
<div class="embed-responsive embed-responsive-16by9">
    <embed width="100%" height="100%" src="http://www.tudou.com/v/OUG5JBZ8udc/&bid=05&rpi
</div>
```



//4:3 响应式

```
<div class="embed-responsive embed-responsive-4by3">
    <embed width="100%" height="100%" src="http://www.tudou.com/v/OUG5JBZ8udc/&bid=05&rpi
</div>
```



第 13 章 模态框插件

学习要点：

1.基本使用

2.用法说明

主讲教师：李炎恢

本节课我们主要学习一下 Bootstrap 中的模态框插件， 这是一款交互式网站非常常见的弹窗功能插件。

一．基本使用

使用模态框的弹窗组件需要三层 div 容器元素，分别为 modal(模态声明层)、

dialog(窗口声明层)、content(内容层)。在内容层里面，还有三层，分别为 header(头部)、body(主体)、footer(注脚)。

//基本实例

```
<!-- 模态声明，show 表示显示 -->

&lt;div class="modal show" tabindex="-1"&gt;
    &lt;!-- 窗口声明 --&gt;
    &lt;div class="modal-dialog"&gt;
        &lt;!-- 内容声明 --&gt;
        &lt;div class="modal-content"&gt;
            &lt;!-- 头部 --&gt;
            &lt;div class="modal-header"&gt;
                &lt;button type="button" class="close" data-dismiss="modal"&gt;
                    &lt;span&gt;&times;&lt;/span&gt;
                &lt;/button&gt;
                &lt;h4 class="modal-title"&gt;会员登录&lt;/h4&gt;
            &lt;/div&gt;
            &lt;!-- 主体 --&gt;
            &lt;div class="modal-body"&gt;
                &lt;p&gt; 暂时无法登录会员 &lt;/p&gt;
            &lt;/div&gt;
            &lt;!-- 注脚 --&gt;
            &lt;div class="modal-footer"&gt;
                &lt;button type="button" class="btn btn-default"&gt; 注册 &lt;/button&gt;
                &lt;button type="button" class="btn btn-primary"&gt; 登录 &lt;/button&gt;
            &lt;/div&gt;
        &lt;/div&gt;
    &lt;/div&gt;
&lt;/div&gt;
```

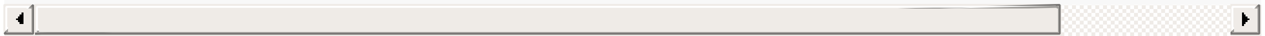
如果想让模态框自动隐藏，然后通过点击按钮弹窗，那么需要做如下操作。

//模态框去掉 show，增加一个 id

```
<div class="modal" id="myModal">
```

//点击触发模态框显示

```
<button class="btn btn-primary btn-lg" data-toggle="modal" data-target="#myModal"> 点击弹窗
```



//弹窗的大小有三种，默认情况下是正常，还有 lg(大)和 sm(小)

```
<div class="modal-dialog modal-lg">
<div class="modal-dialog sm-lg">
```

//可设置淡入淡出效果

```
<div class="modal fade" id="myModal">
```

//在主体部分使用栅格系统中的流体

```
<!-- 主体 -->

&lt;div class="modal-body"&gt;
    &lt;div class="container-fluid"&gt;
        &lt;div class="row"&gt;
            &lt;div class="col-md-4"&gt; 1 &lt;/div&gt;
            &lt;div class="col-md-4"&gt; 1 &lt;/div&gt;
            &lt;div class="col-md-4"&gt; 1 &lt;/div&gt;
        &lt;/div&gt;
    &lt;/div&gt;
&lt;/div&gt;
```

二．用法说明

基本使用介绍结束之后，我们就来看下插件的各种重要用法。所有的插件，都是基于 JavaScript/jQuery 的。那么，就有四个要素：用法、参数、方法和事件。

1.用法

第一种：可以通过 data 属性

//data-toggle

```
data-toggle="modal" data-target="#myModal"
```

data-toggle 表示触发类型

data-target 表示触发的节点

如果不是使用<button>，而是<a>，其中 data-target 也可以使用 href="#myModal"

取代。当然，我们建议使用 data-target。除了 data-toggle 和 data-target 两个声明属性外，还有一些可以用选项。

2.参数

可以通过在 HTML 元素上设置 data-*的属性声明来控制效果。

属性名称	类型	默认值	描述
data-backdrop	布尔值或'static'	true	默认值 true，表示背景存在黑灰透明遮罩，且单击空白背景可关闭弹窗；如果为 false，表示背景不存在黑灰透明遮罩，且点击空白背景不可关闭弹窗；如果是字符串'static'，表示背景存在黑灰透明遮罩，且点击空白不可关闭弹窗。
data-keyboard	布尔值	true	如果是 true，按 esc 键会关闭窗口；如果是 false，按 esc 键会不会关闭。
data-show	布尔值	true	如果是 true，初始化时，默认显示；如果是 false，初始化时，默认隐藏。
href	url 路径	空值	如果值不是以#号开头，则表示一个 url 地址，加载 url 内容到 modal-content 容器里，并只加载一次。如果是#号，就是取代 data-target 的方法。

//空白背景且点击不关闭

```
data-backdrop="false"
```

//按下 esc 不关闭

```
data-keyboard="false"
```

//初始化隐藏，如果是按钮点击触发，第一次点击则无法显示，第二次显示。

```
data-show="false"
```

//加载一次 index.html 到容器内

```
href="index.html"
```

当然，也可以在 JavaScript 直接设置。

属性名称	类型	默认值	描述
backdrop	布尔值或 'static'	true	默认值 true，表示背景存在黑灰透明遮罩，且单击空白背景可关闭弹窗；如果为 false，表示背景不存在黑灰透明遮罩，且点击空白背景不可关闭弹窗；如果是字符串'static'，表示背景存在黑灰透明遮罩，且点击空白不可关闭弹窗。
keyboard	布尔值	true	如果是 true，按 esc 键会关闭窗口；如果是 false，按 esc 键会不会关闭。
show	布尔值	true	如果是 true，初始化时，默认显示；如果是 false，初始化时，默认隐藏。
remote	url 路径	空值	远程获取指定内容填充到 modal-content 容器内。

//通过 jQuery 方式声明

```
$('#myModal').modal({
    show : true,
    backdrop : false,
    keyboard : false,
    remote : 'index.html',
});
```

3.方法

如果说，默认不显示弹窗，那么怎么才能通过点击前后弹窗呢？

参数名称	使用方法	描述
toggle	.modal('toggle');	触发时，反转切换弹窗状态
show	.modal('show');	触发时，显示弹窗
hide	.modal('hide');	触发时，关闭弹窗

//点击显示弹窗

```
$('#btn').on('click', function() {
    $('#myModal').modal('show');
});
```

4.事件

模态框支持 4 种时间，分别对应弹出前、弹出后、关闭前和关闭后。

事件类型	描述
show.bs.modal	在 show 方法调用时立即触发。
shown.bs.modal	在模态框完全显示出来，并且等 CSS 动画完成之后触发。
hide.bs.modal	在 hide 方法调用时，但还未关闭隐藏时触发。
hidden.bs.modal	在模态框完全隐藏之后，并且等 CSS 动画完成之后触发。

```
$('#myModal').on('show.bs.modal', function() {
    alert('在 show 方法调用时立即触发！');
});

$('#myModal').on('shown.bs.modal', function() {
    alert('在模态框显示完毕后触发！');
});

$('#myModal').on('hide.bs.modal', function() {
    alert('在 hide 方法调用时立即触发！');
});

$('#myModal').on('hiden.bs.modal', function() {
    alert('在模态框显示完毕后触发！');
});

$('#myModal').on('loaded.bs.modal', function() {
    alert('远程数据加载完毕后触发！');
});
```

第 14 章 下拉菜单和滚动监听插件

学习要点：

1.下拉菜单

2.滚动监听

主讲教师：李炎恢

本节课我们主要学习一下 Bootstrap 中的下拉菜单插件， 这个插件在以组件的形式我们已经学习过，那么现在来看看怎么和 JavaScript 交互的。

一．下拉菜单

常规使用中，和组件方法一样，代码如下：

//声明式用法

```
<div class="dropdown">
    <button class="btn btn-primary" data-toggle="dropdown"> 下拉菜单 <span class="caret"><
    </button>
    <ul class="dropdown-menu">
          <li><a href="#">首页</a></li>
          <li><a href="#">产品</a></li>
          <li><a href="#">资讯</a></li>
          <li><a href="#">关于</a></li>
    </ul>
</div>
```

声明式用法的关键核心：

1.外围容器使用 class="dropdown"包裹；

2.内部点击按钮事件绑定 data-toggle="dropdown"；

3.菜单元素使用 class="dropdown-menu"。

//如果按钮在容器外部，可以通过 data-target 进行绑定。

```
<button class="btn btn-primary" id="btn" data-toggle="dropdown" data-target="#dropdown">
```

在 JavaScript 调用中，没有属性，方法并不好用，下面介绍四个基本事件。

//下拉菜单方法，但仍然需要 data-*

```
$('#btn').dropdown();
$('#btn').dropdown('toggle');
```

下拉菜单支持 4 种事件，分别对应弹出前、弹出后、关闭前和关闭后。

事件类型	描述
show.bs.dropdown	在 show 方法调用时立即触发。
shown.bs.dropdown	在下拉菜单完全显示出来，并且等 CSS 动画完成之后触发。
hide.bs.dropdown	在 hide 方法调用时，但还未关闭隐藏时触发。
hidden.bs.dropdown	在下拉菜单完全隐藏之后，并且等 CSS 动画完成之后触发。

//事件，其他雷同

```
$('#dropdown').on('show.bs.dropdown', function() {
    alert('在调用 show 方法时立即触发！');
});
```

二．滚动监听

滚动监听插件是用来根据滚动条所处在的位置自动更新导航项目， 显示导航项目高亮显示。

//基本实例

```
<nav id="nav" class="navbar navbar-default">
    <a href="#" class="navbar-brand">Web 开发</a>
    <ul class="nav navbar-nav">
        <li>
            <a href="#html5">HTML5</a>
        </li>
        <li>
            <a href="#bootstrap">Bootstrap</a>
        </li>
        <li class="dropdown">
            <a href="#" data-toggle="dropdown">JavaScript <span class="caret"></span></a>
            <ul class="dropdown-menu">
                <li>
                    <a href="#jquery">jQuery</a>
                </li>
                <li>
                    <a href="#yui">Yui</a>
                </li>
                <li>
                    <a href="#extjs">Extjs</a>
                </li>
            </ul>
        </li>
    </ul>
</nav>

<div data-offset="0" data-target="#nav" data-spy="scroll" style="height: 200px; overflow:
    <h4 id="html5">HTML5</h4>
    <p> 标准通用标记语言下的一个应用 HTML 标准自 1999 年 12 月发布的 HTML4.01后，后继的 HTML5 和其

        双方决定进行合作，来创建一个新版本的 HTML。 </p>
    <h4 id="bootstrap">Bootstrap</h4>
    <p> Bootstrap，来自 Twitter，是目前很受欢迎的前端框架。Bootstrap 是基于 HTML、 CSS、 JAVASCR
    <h4 id="jquery">jQuery</h4>
    <p> JQuery 是继 prototype 之后又一个优秀的 Javascript 库。 它是轻量级的 js库 ，它兼容 CSS3，这
    <h4 id="yui">Yui</h4>
    <p> 近几年随着 jQuery、Ext 以及 CSS3 的发展，以 Bootstrap 为代表的前端开发框架如雨后春笋般挤入视
    <h4 id="extjs">Extjs</h4>
    <p> ExtJS 可以用来开发 RIA 也即富客户端的 AJAX 应用，是一个用 javascript写的，主要用于创建前端用
</div>
```

这里有两个重要的属性，如下图：

属性名	描述
data-offset	默认值为 10，固定弄内容距滚动容器 10 像素以内，就高亮显示所对应的菜单。
data-spy	设置 scroll，将设置滚动容器监听。
data-target	设置#nav，绑定指定监听的菜单

PS：在一个菜单和一个容易的时候，data-target 不设置也可以稳定实现滚动监听高亮。但多个导航时，你不关联其中一个，会导致错误，所以，一般要加上。

如果使用 JavaScript 脚本方式， 可以去掉 data-*， 使用脚本属性定义： offset、 spy和 target。具体方法如下：

//使用脚本方式定义属性

```
$('#content').scrollspy({
    offset : 0,
    target : '#nav',
});
```

滚动监听还有一个切换到新条目的事件。

事件名	描述
activate.bs.scrollspy	每当一个新条目被激活后都将由滚动监听插件触发此事件。

//事件绑定在导航上

```
$('#nav').on('activate.bs.scrollspy', function() {
    alert('新条目被激活后触发此事件！');
});
```

滚动监听还有一个更新容器 DOM 的方法。

方法名	描述
refresh	更新容器 DOM 的方法。

//HTML 部分

```
<section class="sec">
    <h4 id="html5">HTML5 <a href="#" onclick="removeSec(this)">删除此项</a></h4>
    <p> ... </p>
</section>
```

//删除内容时，刷新一下 DOM，避免导航监听错位

```
function removeSec(e) {
    $(e).parents('.sec').remove();
    $('#content').scrollspy('refresh');
}
```

注意：这个方法必须使用 data-*声明式。

第 15 章 标签页和工具提示插件

学习要点：

1.标签页

2.工具提示

主讲教师：李炎恢

本节课我们主要学习一下 Bootstrap 中的标签页和工具提示插件。

一．标签页

标签页也就是通常所说的选项卡功能。

//基本用法

```
<ul class="nav nav-tabs">
    <li class="active">
        <a href="#html5" data-toggle="tab">HTML5</a>
    </li>
    <li>
        <a href="#bootstrap" data-toggle="tab">Bootstrap</a>
    </li>
    <li>
        <a href="#jquery" data-toggle="tab">jQuery</a>
    </li>
    <li>
        <a href="#extjs" data-toggle="tab">ExtJS</a>
    </li>
</ul>

<div class="tab-content" style="padding: 10px;">
    <div class="tab-pane active" id="html5"> ... </div>
    <div class="tab-pane" id="bootstrap"> ... </div>
    <div class="tab-pane" id="jquery"> ... </div>
    <div class="tab-pane" id="extjs"> ... </div>
</div>
```

//可以设置淡入淡出效果 fade，而 in 表示首选的内容默认显示

```
<div class="tab-pane fade in active" id="html5">
```

//也可以换成胶囊式

```
<ul class="nav nav-pills">
```

//data-target

使用 data-target 绑定或不绑定效果都是一样的

//使用 JavaScript，直接使用 tab 方法。

```
$('#nav a').on('click', function(e) {
    e.preventDefault();
    $(this).tab('show');
});
```

事件类型	描述
show.bs.tab	在调用 tab 方法时触发
shown.bs.tab	在显示整个标签时触发

//事件，其他雷同

```
$('#nav a').on('show.bs.tab', function() {
    alert('调用 tab 时触发！');
});

$('#nav a').on('shown.bs.tab', function() {
    alert('显示完 tab 时触发！');
});
```

二．工具提示

工具提示就是通过鼠标移动选定在特定的元素上时，显示相关的提示语。

//基本实例

```
<a href="#" data-toggle="tooltip" title="超文本标识符">HTML5</a>
```

//JS 部分需要声明

```
$('#section').tooltip();
```

工具提示有很多属性来配置提示的显示，具体如下：

属性名	描述
data-animation	默认 true，在 tooltip 上应用一个 CSS fade 动画。如果设置 false，则不应用。
data-html	默认 false，不允许提示内容格式为 html。如果设置为 true，则可以设置 html 格式的提示内容。
data-placement	默认值 top，还有 bottom、left、right 和 auto。如果 auto 会自行调整合适的位置，如果是 auto left 则会尽量在左边显示，但左边不行就靠右边。
data-selector	默认 false，可以选择绑定指定的选择器。
data-original-title	默认空字符串，提示语的内容。优先级比 title 低
title	默认字空符串，提示语的内容。
data-trigger	默认值 hover foucs，表示怎么触发 tooltip，其他值为：click、manual。多个值用空格隔开，manual 手动不能和其他同时设置。
data-delay	默认值 0，延迟触发 tooltip(毫秒)，如果传数字则，表示 show/hide 的毫秒数，如果传对象，结构为：
	{show:500,hide:100}
data-container	默认值 false，将 tooltip 附加到特定的元素上。比如组合按钮组提示，容器不够，可以附加 body 上。container : 'body'
data-template	更改提示框的 HTML 提示语的模版，默认值为：<div class='tooltip'><div class='tooltip-arrow'></div><div class='tooltip-inner'></div></div>。

```
<a href="#" rel="tooltip" data-toggle="tooltip" title="超文本标识符" data-animation="false"
>HTML5</a>
```



JavaScript 方式直接去掉前面的 data 即可。包括：animation、html、placement、selector、original-title、title、trigger、delay、container 和 template 等属性。

//JavaScript 方式

```
$('#section a').tooltip({
    delay : {
        show : 500,
        hide : 100,
    },
    container : 'body' });
```

JavaScript 有四个方法：show、hide、toggle 和 destroy 四种。

```
//显示
$('#section a').tooltip('show'); //隐藏
$('#section a').tooltip('hide'); //反转显示和隐藏
$('#section a').tooltip('toggle'); //隐藏并销毁
$('#section a').tooltip('destroy');
```

Tooltip 中事件有四种。

事件类型	描述
show.bs.tooltip	在 show 方法调用时立即触发
shown.bs.tooltip	在提示框完全显示给用户之后触发
hide.bs.tooltip	在 hide 方法调用时立即触发
hidden.bs.tooltip	在提示框完全隐藏之后触发

//事件，其他雷同

```
$('#select a').on('show.bs.tooltip', function() {
    alert('调用 show 时触发！');
});
```

第 16 章 弹出框和警告框插件

学习要点：

1.弹出框

2.警告框

主讲教师：李炎恢

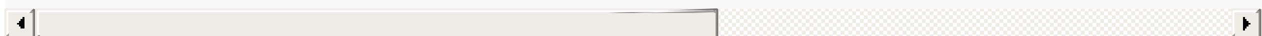
本节课我们主要学习一下 Bootstrap 中的弹出框和警告框插件。

一．弹出框

弹出框即点击一个元素弹出一个包含标题和内容的容器。

//基本用法

```
<button class="btn btn-lg btn-danger" type="button" data-toggle="popover" title="弹出框" d
```



//JavaScript 初始化

```
$('button').popover();
```

弹出框插件有很多属性来配置提示的显示，具体如下：

属性名	描述
data-animation	默认 true，在 popover 上应用一个 CSS fade 动画。如果设置 false，则不应用。
data-html	默认 false，不允许提示内容格式为 html。如果设置为 true，则可以设置 html 格式的提示内容。
data-placement	默认值 top，还有 bottom、left、right 和 auto。如果 auto 会自行调整合适的位置，如果是 auto left 则会尽量在左边显示，但左边不行就靠右边。
data-selector	默认 false，可以选择绑定指定的选择器。
data-original-title	默认空字符串，弹出框的标题。优先级比 title 低
title	默认字空符串，弹出框的标题。
data-trigger	默认值 click，表示怎么触发 popover，其他值为：hover、focus、manual。多个值用空格隔开，manual 手动不能和其他同时设置。
data-delay	默认值 0，延迟触发 popover(毫秒)，如果传数字则，表示 show/hide 的毫秒数，如果传对象，结构为：{show:500,hide:100}
data-container	默认值 false，将 popover 附加到特定的元素上。比如组合按钮组提示，容器不够，可以附加 body 上。container : 'body'
data-template	更改提示框的 HTML 提示语的模版，默认值为：<div class="popover"><div class="arrow"></div><h3 class="popover-title"></h3><div class="popover-content"></div></div>
data-content	默认值为空，弹出框的内容。
data-viewport	设置外围容器的边际，具体代码看示例。

```
$('button').popover({
    container : 'body',
    viewport : {
        selector : '#view',
        padding : 10,
    }
});
```

通过 JavaScript 执行的方法有四个。

```
//显示
$('button').popover('show'); //隐藏
$('button').popover('hide'); //反转显示和隐藏
$('button').popover('toggle'); //隐藏并销毁
$('button').popover('destroy');
```

Popover 插件中事件有四种：

事件类型	描述
show.bs.popover	在调用 show 方法时触发
shown.bs.popover	在显示整个弹窗时时触发
hide.bs.popover	在调用 hide 方法时触发
hidden.bs.popover	在完全关闭整个弹出时触发

//事件，其他雷同

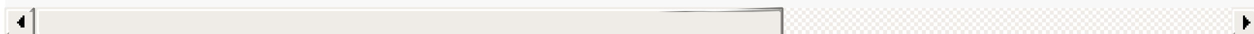
```
$('button').on('show.bs.tab', function() {
    alert('调用 show 方法时触发！');
});
```

二．警告框

警告框即为点击小时的信息框。

//基本实例

```
<div class="alert alert-warning">
    <button class="close" type="button" data-dismiss="alert">
        <span>&times;</span>
    </button>
    <p> 警告：您的浏览器不支持！ </p>
</div>
```



//添加淡入淡出效果

```
<div class="alert alert-warning fade in">
```

如果用 JavaScript，可以代替 data-dismiss="alert"

//JavaScript 方法

```
$('.close').on('click', function() {
    $('#alert').alert('close');
})
```

Alert 插件中事件有两种：

事件类型	描述
close.bs.alert	当 close 方法被调用后立即触发
closed.bs.alert	当警告框被完全关闭后立即触发

//事件，其他雷同

```
$('#alert').on('close.bs.alert', function() {
    alert('当 close 方法被触发时调用！');
});
```

第 17 章 按钮和折叠插件

学习要点：

1.按钮

2.折叠

主讲教师：李炎恢

本节课我们主要学习一下 Bootstrap 中的按钮和折叠插件。

一．按钮

可以通过按钮插件创建不同状态的按钮。

//单个切换。

```
<button class="btn btn-primary" data-toggle="button" autocomplete="off">单个切换</button>
```

注：在 Firefox 多次页面加载时，按钮可能保持表单的禁用或选择状态。解决方案是：添加 autocomplete="off"。

//单选按钮

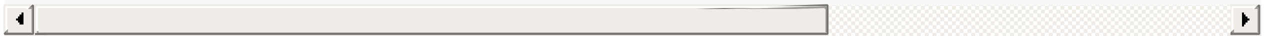
```
<div class="btn-group" data-toggle="buttons">
    <label for="" class="btn btn-primary active">
        <input type="radio" name="sex" autocomplete="off" checked>男 </label>
    <label for="" class="btn btn-primary">
        <input type="radio" name="sex" autocomplete="off">女 </label>
</div>
```

//复选按钮

```
<div class="btn-group" data-toggle="buttons">
    <label for="" class="btn btn-primary active">
        <input type="checkbox" name="fa" autocomplete="off" checked> 音乐 </label>
    <label for="" class="btn btn-primary">
        <input type="checkbox" name="fa" autocomplete="off"> 体育 </label>
    <label for="" class="btn btn-primary">
        <input type="checkbox" name="fa" autocomplete="off"> 美术 </label>
    <label for="" class="btn btn-primary">
        <input type="checkbox" name="fa" autocomplete="off"> 电脑 </label>
</div>
```

//加载状态

```
<button id="myButton" type="button" data-loading-text="Loading..."class="btn btn-primary"
```



```
$('#myButton').on('click', function() { var btn = $(this).button('loading');
    setTimeout(function() {
        btn.button('reset');
    }, 1000);
});
```

Button 插件中的 button 方法中有三个参数： toggle、 reset、 string(比如 loading、complete)。

//可代替 data-toggle="button"

```
$('button').on('click', function() {
    $(this).button('toggle');
})
```

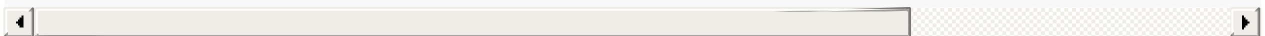
二．折叠

通过点击可以折叠内容。

//基本实例

```
<button class="btn btn-primary" data-toggle="collapse" data-target="#content"> Bootstrap

<div class="collapse" id="content">
    <div class="well"> Bootstrap 是 Twitter 推出的一个用于前端开发的开源工具包。它由
        Twitter 的设计师 Mark Otto 和 Jacob Thornton 合作开发,是一个 CSS/HTML 框架。目前,Bootst
</div>
```



//手风琴折叠

```
<div class="panel-group" id="accordion">
    <div class="panel panel-default">
        <div class="panel-heading">

            <h4 class="panel-title"><a href="#collapseOne" data-toggle="collapse" data-pa
        </div>
        <div id="collapseOne" class="panel-collapse collapse in">
            <div class="panel-body"> 这里是第一部分。 </div>
        </div>
    </div>

    <div class="panel panel-default">
        <div class="panel-heading">
            <h4 class="panel-title"><a href="#collapseTwo" data-toggle="collapse" data-pa
        </div>
        <div id="collapseTwo" class="panel-collapse collapse">
            <div class="panel-body"> 这里是第二部分。 </div>
        </div>
    </div>

    <div class="panel panel-default">
        <div class="panel-heading">
            <h4 class="panel-title"><a href="#collapseThree" data-toggle="collapse" data-
        </div>
        <div id="collapseThree" class="panel-collapse collapse">
            <div class="panel-body"> 这里是第三部分。 </div>
        </div>
    </div>
</div>
```

属性名称	描述
data-parent	默认值为 false，设置需指定父元素选择器。也就是说，选定其中一个折叠区，其他折叠将隐藏，实现手风琴效果。
data-toggle	如果前面加 data-*，设置'collapse'表示实现折叠；如果是 JavaScript 中的属性，默认为 true，实现反转。

//手风琴效果

```
$('#collapseOne, #collapseTwo,#collapseThree, #collapseFour').collapse({
    parent : '#accordion',
    toggle : false,
});
```

//手动调用

```
$('button').on('click', function() {
    $('#collapseOne').collapse({
        toggle : true,
    });
});
```

//collapse 方法还提供了三个参数：hide、show、toggle。

```
$('#collapseOne').collapse('hide');
$('#collapseTwo').collapse('show');
$('button').on('click', function() {
    $('#collapseOne').collapse('toggle');
});
```

Collapse 插件中事件有四种。

事件类型	描述
show.bs.collapse	在 show 方法调用时立即触发
shown.bs.collapse	折叠区完全显示出来是触发
hide.bs.collapse	在 hide 方法调用时触发
hidden.bs.collapse	该事件在折叠区域完全隐藏之后触发

//事件，其他雷同

```
$('#collapseOne').on('show.bs.collapse', function() {
    alert('当 show 方法调用时触发');
});
```

第 18 章 轮播插件

学习要点：

1.轮播插件

主讲教师：李炎恢

本节课我们主要学习一下 Bootstrap 中的轮播插件。

一．轮播

轮播插件就是将几张同等大小的大图，按照顺序依次播放。

//基本实例。

```
<div id="myCarousel" class="carousel slide">
    <ol class="carousel-indicators">
        <li data-target="#myCarousel" data-slide-to="0" class="active"></li>
        <li data-target="#myCarousel" data-slide-to="1"></li>
        <li data-target="#myCarousel" data-slide-to="2"></li>
    </ol>
    <div class="carousel-inner">
        <div class="item active">
            <img src="img/slide1.png" alt="第一张">
        </div>
        <div class="item">
            <img src="img/slide2.png" alt="第二张">
        </div>
        <div class="item">
            <img src="img/slide3.png" alt="第三张">
        </div>
    </div>

    <a href="#myCarousel" data-slide="prev" class="carousel-controlleft">&lsaquo;</a>
    <a href="#myCarousel" data-slide="next" class="carousel-controlright">&rsaquo;</a>
</div>
```

data 属性解释：

1.data-slide 接受关键字 prev 或 next，用来改变幻灯片相对于当前位置的位置；

2.data-slide-to 来向轮播底部创建一个原始滑动索引， data-slide-to="2"将把滑动块移动到一个特定的索引，索引从 0 开始计数。

3.data-ride="carousel"属性用户标记轮播在页面加载时开始动画播放。

属性名称	描述
data-interval	默认值 5000，幻灯片的等待时间(毫秒)。如果为 false，轮播将不会自动开始循环。
data-pause	默认鼠标停留在幻灯片区域(hover)即暂停轮播，鼠标离开即启动轮播。
data-wrap	默认值 true，轮播是否持续循环。

如果在 JavaScript 调用就直接使用键值对方法，并去掉 data-；

//设置自定义属性

```
$('#myCarousel').carousel({ //设置自动播放/2 秒
    interval : 2000, //设置暂停按钮的事件
    pause : 'hover', //只播一次
    wrap : false,
});
```

轮播插件还提供了一些方法，如下：

方法名称	描述
cycle	循环各帧(默认从左到右)
pause	停止轮播
number	轮播到指定的图片上(小标从 0 开始，类似数组)
prev	循环轮播到上一个项目
next	循环轮播到下一个项目

//点击按钮执行

```
$('button').on('click', function() { //点击后，自动播放
    $('#myCarousel').carousel('cycle'); //其他雷同
});
```

方法名称	描述
slide.bs.carousel	当调用 slide 实例方式时立即触发该事件。
slid.bs.carousel	当轮播完成一个幻灯片触发该事件。

//事件

```
$('#myCarousel').on('slide.bs.carousel', function() {
    alert('当调用 slide 实例方式时立即触发');
});

$('#myCarousel').on('slid.bs.carousel', function() {
    alert('当轮播完成一个幻灯片触发');
});
```

第 19 章 附加导航插件

学习要点：

1.附加导航插件

主讲教师：李炎恢

本节课我们主要学习一下 Bootstrap 中的附加导航插件。

一．附加导航

附加导航即粘贴在屏幕某处实现锚点功能。

//基本实例

```
<body data-spy="scroll" data-target="#myScrollspy">

    <div class="container">
        <div class="jumbotron" style="height:150px">
            <h1>Bootstrap Affix</h1>
        </div>
        <div class="row">
            <div class="col-xs-3" id="myScrollspy">
                <ul class="nav nav-pills nav-stacked" data-spy="affix"data-offset-top="15
                    <li class="active">
                        <a href="#section-1">第一部分 </a>
                    </li>
                    <li>
                        <a href="#section-2">第二部分</a>
                    </li>
                    <li>
                        <a href="#section-3">第三部分</a>
                    </li>
                    <li>
                        <a href="#section-4">第四部分</a>
                    </li>
                    <li>
                        <a href="#section-4">第五部分</a>
                    </li>
                </ul>
            </div>
            <div class="col-xs-9">
                <h2 id="section-1">第一部分</h2>
                <p> ... </p>
                <h2 id="section-2">第二部分</h2>
                <p> ... </p>
                <h2 id="section-3">第三部分</h2>
                <p> ... </p>
                <h2 id="section-4">第四部分</h2>
                <p> ... </p>
                <h2 id="section-5">第四部分</h2>
                <p> ... </p>
            </div>

        </div>
    </div>
```

//导航的 CSS 部分

```
ul.nav-pills { width: 200px;
} ul.nav-pills.affix { top: 30px;
}
```

//JavaScript 代替 data-spy="affix" data-offset-top="125"

```
$('#myAffix').affix({
    offset : {
        top : 150 }
})
```

我们默认使用的是 top，当然也可以默认居底 bottom。这个定位方式是直接通过 CSS定位的。

//设置成 bottom

```
ul.nav-tabs.affix-bottom { bottom: 30px;
}
```

//设置成 bottom

```
$('#myAffix').affix({
    offset : {
        bottom : 150 }
})
```

Affix 包含几个事件，如下：

事件名称	描述
affix.bs.affix	在定位结束之前立即触发
affixed.bs.affix	在定位结束之后立即触发
affix-top.bs.affix	在定位元素应用 affixed-top 效果之前触发
affixed-top.bs.affix	在定位元素应用 affixed-top 效果之后触发
affix-bottom.bs.affix	在定位元素应用 affixed-bottom 效果之前触发
affixed-bottom.bs.affix	在定位元素应用 affixed-bottom 效果之后触发

//其他雷同

```
$('#myAffix').on('affixed-top.bs.affix', function() {
    alert('触发！');
});
```

第 20 章 项目实战--响应式导航[1]

学习要点：

1.响应式导航

主讲教师：李炎恢

本节课我们开始设计第一个项目，一个内训公司的企业网站，本节课学习响应式导航部分。

一．响应式导航

//基本导航组件+响应式

```
<nav class="navbar navbar-default navbar-fixed-top">
    <div class="container">
        <div class="navbar-header">
            <a href="#" class="navbar-brand" style="margin:0;padding:0;"><img src="img/lo
            <button type="button" class="navbar-toggle" data-toggle="collapse" data-targe
                <span class="sr-only">切换导航</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
        </div>
        <div class="collapse navbar-collapse" id="navbar-collapse">
            <ul class="nav navbar-nav navbar-right" style="margin-top:0;">
                <li class="active">
                    <a href="#"><span class="glyphiconglyphicon-home"></span> 首页</a>
                </li>
                <li>
                    <a href="#"><span class="glyphicon
                    glyphicon-list"></span> 资讯</a>
                </li>
                <li>
                    <a href="#"><span class="glyphicon
                    glyphicon-fire"></span> 案例</a>
                </li>
                <li>
                    <a href="#"><span class="glyphicon
                    glyphicon-question-sign"></span> 关于</a>
                </li>
            </ul>
        </div>
    </div>
</nav>
```

第 20 章 项目实战--响应式轮播图[2]

学习要点：

1.响应式轮播图

主讲教师：李炎恢

本节课我们要在导航条的下方做一张轮播图，自动播放最新的重要动态。

一．响应式轮播图

//响应式轮播图

```
<div id="myCarousel" class="carousel slide">
    <ol class="carousel-indicators">
        <li data-target="#myCarousel" data-slide-to="0" class="active"></li>
        <li data-target="#myCarousel" data-slide-to="1"></li>
        <li data-target="#myCarousel" data-slide-to="2"></li>
    </ol>
    <div class="carousel-inner">
        <div class="item active" style="background:#223240;">
            <a href="#"><img src="img/slide1.png" alt="第一张"></a>
        </div>
        <div class="item" style="background:#F5E4DC;">
            <a href="#"><img src="img/slide2.png" alt="第二张"></a>
        </div>
        <div class="item" style="background:#DE2A2D;">
            <a href="#"><img src="img/slide3.png" alt="第三张"></a>
        </div>
    </div>
    <a href="#myCarousel" data-slide="prev" class="carousel-controlleft">&lsaquo;</a>
    <a href="#myCarousel" data-slide="next" class="carousel-controlright">&rsaquo;</a>
</div>
```

//所需要的 jQuery 控制

```
$('#myCarousel').carousel({ //设置自动播放/2 秒
    interval : 3000,
});
```

//调整轮播器箭头位置

```
$('.carousel-control').css('line-height', $('.carousel-innerimg').height() + 'px');
$(window).resize(function() { var $height = $('.carousel-inner img').eq(0).height() || $(
    $('.carousel-control').css('line-height', $height + 'px');
});
```



//所需要的 CSS

```
a:focus { outline: none;
} .navbar-brand { padding: 0;
} #myCarousel { margin: 50px 0 0 0;
} .carousel-inner .item img { margin: 0 auto;
} .carousel-control { font-size: 100px;
}
```

第 20 章 项目实战--首页内容介绍[上][3]

学习要点：

1.首页内容介绍[上]

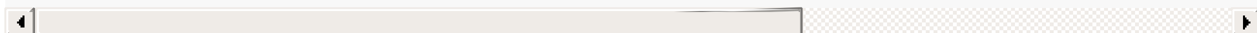
主讲教师：李炎恢

本节课我们轮播图的下方，设计一个内容介绍，内容介绍分两部分，本次为上半部分。

一．首页内容介绍[上]

//关于上节课轮播图，手册上其实有一个更好的方案，并不需要通过额外的代码控制。

```
<a href="#myCarousel" data-slide="prev" class="carousel-control left"> <span class="glyph
<a href="#myCarousel" data-slide="next" class="carousel-controlright"> <span class="glyph
```



//内容介绍上

```
<div class="tab1">
    <div class="container">
        <h2 class="tab-h2">「 为什么选择瓢城企业培训 」</h2>
        <p class="tab-p"> 强大的师资力量，完美的实战型管理课程，让您的企业实现质的腾飞！ </p>
        <div class="row">
            <div class="col-md-6 col">
                <div class="media">
                    <div class="media-left media-top">
                        <a href="#"> <img class="media-object" src="img/tab1-1.png" alt="
                    </div>
                    <div class="media-body">
                        <h4 class="media-heading">课程内容</h4>
                        <p class="text-muted"> 其他：高校不知名的讲师编写，没有任何实战价值的教材
                        <p> 其他：知名企业家、管理学大师联合编写的具有实现性教材！ </p>
                    </div>
                </div>

            </div>
            <div class="col-md-6 col">
                <div class="media">
                    <div class="media-left media-top">
                        <a href="#"> <img class="media-object" src="img/tab1-2.png" alt="
                    </div>
                    <div class="media-body">
                        <h4 class="media-heading">师资力量</h4>
                        <p class="text-muted"> 其他：非欧美正牌大学毕业的、业界没有知名度的讲师！
                        <p> 其他：美国哈佛、耶鲁等世界一流高校、享有声誉的名牌专家！ </p>
                    </div>
                </div>
            </div>
            <div class="col-md-6 col">
                <div class="media">
                    <div class="media-left media-top">
                        <a href="#"> <img class="media-object" src="img/tab1-3.png" alt="
                    </div>
                    <div class="media-body">
                        <h4 class="media-heading">课时安排</h4>
                        <p class="text-muted"> 其他：无法保证上课效率、没有时间表，任务无法完成！
                        <p> 其他：保证正常的上课效率、制定一张时间表、当天的任务当天完成！ </p>
                    </div>
                </div>
            </div>
            <div class="col-md-6 col">
                <div class="media">
                    <div class="media-left media-top">
                        <a href="#"> <img class="media-object" src="img/tab1-4.png" alt="
                    </div>
                    <div class="media-body">

                        <h4 class="media-heading">服务团队</h4>
                        <p class="text-muted"> 其他：社会招聘的、服务水平参差不齐的普通员工！ </p
                        <p> 其他：内部培养、经受过良好高端服务培训的高标准员工！ </p>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

//对应的 CSS 部分

```
body { font-family: "Helvetica Neue", Helvetica, Arial, "Microsoft YaheiUI", "Microsoft Y
} .tab-h2 { font-size: 20px; color: #0059B2; text-align: center; letter-spacing: 1px;
} .tab-p { font-size: 15px; color: #999; text-align: center; letter-spacing: 1px; margin:
} .tab1 { margin: 30px 0; color: #666;
} .tab1 .media-heading { margin: 5px 0 20px 0;
} .tab1 .text-muted { color: #999; text-decoration: line-through;
} .tab1 .media-heading { margin: 5px 0 20px 0;
} .tab1 .text-muted { color: #999; text-decoration: line-through;
} .tab1 .col { padding: 20px;
}

/* 小屏幕（平板，大于等于 768px） */ @media (min-width: 768px) { .tab-h2 {
        font-size: 26px;
    } .tab-p { font-size: 16px;
    } } /* 中等屏幕（桌面显示器，大于等于 992px） */ @media (min-width: 992px) { .tab-h2 {
        font-size: 28px;
    } .tab-p { font-size: 17px;
    } } /* 大屏幕（大桌面显示器，大于等于 1200px） */ @media (min-width: 1200px) { .tab-h2 {
        font-size: 30px;
    } .tab-p { font-size: 18px;
    } }
```

第 20 章 项目实战--首页内容介绍[下][4]

学习要点：

1.首页内容介绍[下]

主讲教师：李炎恢

本节课我们制作一下首页内容介绍的下半部分。

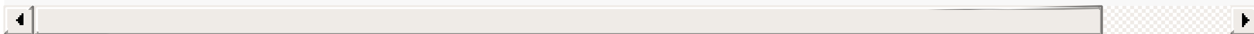
一．首页内容介绍[下]

//先完成底部的 footer

```
<footer id="footer" class="text-muted">
    <div class="container">
        <p> 企业培训 | 合作事宜 | 版权投诉 </p>
        <p> 苏 ICP 备 12345678\. © 2009-2016 瓢城企训网. Powered by
            Bootstrap. </p>
    </div>
</footer>
```

//底部 CSS

```
#footer { padding: 20px; text-align: center; background-color: #eee; border-top: 1px soli
}
```



//两段内容

```
<div class="tab2">
    <div class="container">
        <div class="row">
            <div class="col-md-6 col-sm-6 tab2-img">
                <img src="img/tab2.png" alt="" class="auto
                img-responsive center-block">
            </div>
            <div class="text col-md-6 col-sm-6 tab2-text">
                <h3>强大的学习体系</h3>
                <p> 经过管理学大师层层把关、让您的企业突飞猛进。 </p>
            </div>
        </div>
    </div>
</div>

<div class="tab3">
    <div class="container">
        <div class="row">
            <div class="col-md-6 col-sm-6">
                <img src="img/tab3.png" alt="" class="auto
                img-responsive center-block">
            </div>
            <div class="text col-md-6 col-sm-6">
                <h3>完美的管理方式</h3>
                <p> 最新的管理培训方案，让您的企业赶超同行。 </p>
            </div>
        </div>
    </div>
</div>
```

//CSS 部分

```
.tab2 { background: #eee; padding: 60px 20px; text-align: center;
} .tab2 img { width: 40%; height: 40%;
} .tab3 { padding: 40px 0; text-align: center;
} .tab3 img { width: 65%; height: 65%;
} .text h3 { font-size: 20px;
} .text p { font-size: 14px;
}

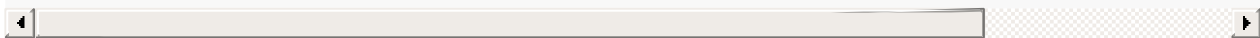
/* 小屏幕（平板，大于等于 768px） */ @media (min-width: 768px) { .text h3 {

        font-size: 22px;
    } .text p { font-size: 15px;
    } .tab2-text { float: left;
    } .tab2-img { float: right;
    } } /* 中等屏幕（桌面显示器，大于等于 992px） */ @media (min-width: 992px) { .text h3 {
        font-size: 24px;
    } .text p { font-size: 16px;
    } .tab2-text { float: left;
    } .tab2-img { float: right;
    } } /* 大屏幕（大桌面显示器，大于等于 1200px） */ @media (min-width: 1200px) { .text h2 {
        font-size: 26px;
    } .text p { font-size: 18px;
    } .tab2-text { float: left;
    } .tab2-img { float: right;
    } }
```

//JS 控制垂直居中

```
$('.text').eq(0).css('margin-top', ($('.auto').eq(0).height() - $('.text').eq(0).height()
$(window).resize(function() {
    $('.text').eq(0).css('margin-top', ($('.auto').eq(0).height() - $('.text').eq(0).heig
});

$('.text').eq(1).css('margin-top', ($('.auto').eq(1).height() - $('.text').eq(1).height()
$(window).resize(function() {
    $('.text').eq(1).css('margin-top', ($('.auto').eq(1).height() - $('.text').eq(1).heig
});
```



第 20 章 项目实战--资讯内容[5,6]

学习要点：

1.资讯内容

主讲教师：李炎恢

本节课我们制作一下子栏目资讯内容。

一．资讯内容

//谷歌浏览器解析的顺序调整，需要全部加载后执行

```
$(window).load(function() {
    $('.text').eq(0).css('margin-top', ($('.auto').eq(0).height() - $('.text').eq(0).heig
});
```

注：对于 Firefox 浏览器，可以按 Ctrl+Shift+M，调整移动端尺寸。

//子栏目标题

```
<div class="jumbotron">
    <div class="container">
        <hgroup>
            <h1>资讯</h1>
            <h4>企业内训的最新动态、资源等...</h4>
        </hgroup>
    </div>
</div>
```

//栏目 CSS

```
.jumbotron { margin: 50px 0 0 0; padding: 60px 0; background: #ccc url(../img/bg.jpg); co
} .jumbotron h1 { font-size: 26px;//768,30; 992,33; 1200,36;
    padding: 0 0 0 20px;
} .jumbotron h4 { font-size: 16px;//768,16; 992,17; 1200,18
    padding: 0 0 0 20px;
}
```

//资讯内容

```
<div id="information">
    <div class="container">
        <div class="row">
            <div class="col-md-8 info-left">
                <div class="container-fluid" style="padding:0;">
                    <div class="row info-content">
                        <div class="col-md-5 col-sm-5 col-xs-5">
                            <img src="img/info1.jpg" class="img-responsive" alt="">
                        </div>
                        <div class="col-md-7 col-sm-7 col-xs-7">
                            <h4>广电总局发布 TVOS2.0 华为阿里参与研发</h4>
                            <p class="hidden-xs"> TVOS2.0 是在 TVOS1.0 与华为 MediaOS 及阿里
                            <p> admin 15 / 10 / 11 </p>
                        </div>
                    </div>
                </div>
            </div>
            <div class="col-md-4 info-right hidden-xs hidden-sm">
                <blockquote>
                    <h2>热门资讯</h2>
                </blockquote>
                <div class="container-fluid">
                    <div class="row">
                        <div class="col-md-5 col-sm-5 col-xs-5" style="margin:12px 0;padd
                            <img src="img/info3.jpg" class="img-responsive" alt="">
                        </div>
                        <div class="col-md-7 col-sm-7 col-xs-7" style="padding-right:0">
                            <h4>标题</h4>
                            <p> admin 15 / 10 / 11 </p>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
```

//资讯内容 CSS

```
#information { padding: 40px 0; background: #eee;
} .info-right { background-color: #fff; box-shadow: 2px 2px 3px #ccc;
} .info-right blockquote { padding: 0; margin: 0;
} .info-right h2 { font-size: 20px; padding: 5px;
} .info-right h4 { line-height: 1.6;
} .info-content { background-color: #fff; box-shadow: 2px 2px 3px #ccc; margin: 0 0 20px
} .info-content img { margin: 12px 0;
} .info-content h4 { font-size: 14px;//768,16; 992,18; 1200,20;
    padding: 2px 0 0 0;
} .info-content p { line-height: 1.6; color: #666;
}
```

//对于.info-content h4，在中屏和大屏需要保持一行。

```
.info-content h4 { overflow: hidden; white-space: nowrap; text-overflow: ellipsis;
}
```

第 20 章 项目实战--案例和关于[7]

学习要点：

1.案例内容

2.关于内容

主讲教师：李炎恢

本节课我们制作一下子案例栏目和关于栏目。

一．案例内容

//案例内容 1-4 个根据不同显示比例展示

```
<div class="col-lg-3 col-md-4 col-sm-6 col-xs-12 col">
    <div class="thumbnail">
        <img src="img/case1.jpg" alt="">
        <div class="caption">
            <h4>中国移动通信</h4>
            <p> 参与了本机构的总裁管理培训课程，学员反馈意见良好。 </p>
        </div>
    </div>
</div>
```

//CSS 部分

```
#case { padding: 40px 0; text-align: center; background-color: #eee;
} #case h4 { color: #666;
} #case p { color: #666; line-height: 1.6;
} #case .col { margin: 0 0 20px 0;
}
```

二．关于栏目

//左右两栏即可

```
<div class="row">
    <div class="col-md-3 hidden-sm hidden-xs">
        <div class="list-group">
            <a class="list-group-item" href="#1">1.机构简介</a>
            <a class="list-group-item" href="#2">2.加入我们</a>
            <a class="list-group-item" href="#3">3.联系方式</a>
        </div>
    </div>
    <div class="col-md-9 about">
        <a name="1"></a>
        <h3>机构简介</h3>
        <p> 瓢城企业培训有限公司是一家专业以智能化弱电工程为主的高科技民营企业，公司自创立以来一直专业致
        <a name="2"></a>
        <h3>加入我们</h3>
        <p> 网络已深刻改变着人们的生活，本地化生活服务市场前景巨大，生活半径团队坚信本地化生活服务与互联
        <p> 请发送您的简历到：hr@xxx.com，我们会在第一时间联系您！ </p>
        <a name="3"></a>
        <h3>联系方式</h3>
        <p> 地址：江苏省盐城市亭湖区大庆中路 1234 号 </p>
        <p> 邮编：1234567 </p>
        <p> 电话：010-88888888 </p>
        <p> 传真：010-88666666 </p>
    </div>
</div>
```

//CSS 部分

```
#about { padding: 40px 15px; background-color: #eee;
} #about .about { background-color: #fff; box-shadow: 2px 2px 3px #ccc; padding-bottom: 2
} #about h3 { margin: 0 0 10px 0; padding: 20px 0; border-bottom: 1px solid #eee; font-si
```

javaScript 教程

javascript快速入门1--JavaScript前世今生,HelloWorld与开发环境

JavaScript历史

大概在1992年,一家称作Nombas的公司开始开发一种叫做C--(C-minus-minus,简称Cmm)的嵌入式脚本语言。 Cmm背后的理念很简单：一个足够强大可以替代宏操作（macro）的脚本语言，同时保持与C（和C++）中够的相似性，以便开发人员能很快学会。 这个脚本语言捆绑在一个叫做CEnvi的共享软件产品中，它首次向开发人员展示了这种语言的威力。 Nombas最终把Cmm的名字改成了ScripEase。原因是后面的部分（mm）听起来过于“消极”，同时字母C“令人害怕”。 现在ScriptEase已经成为了Nombas产品背后的主要驱动力。 当Netscape Navigator崭露头角时，Nombas开发了个可以嵌入网页中的CEnvi的版本。 这些早期的试验称为Espresso Page（浓咖啡般的页面），它们代表了每个在万维网上使用的客户端脚本语言。 而Nombas丝毫没有料到它的理念将会成为因特网的一块重要基石。

当网上冲浪越来越流行时，对于开发客户端脚本的需求也逐渐增大。此时，大部分因特网用户还仅仅通过28.8kbit/s的调制解调器来连接到网络，即便这时网页已经不断地变得更大和更复杂。而更加加剧用户痛苦的是，仅仅为了简单的表单有效性验证，就要与服务器端进行多次的往返交互。设想一下，用户填完一个表单，点击提交按钮，等待了30秒钟的处理后，看到的却是一条告诉你忘记填写一个必要的字段。那时正处于技术革新最前沿的Netscape，开始认真考虑一种开发客户端脚本语言来解决简单的处理问题。

当时工作于Netscape的Brendan Eich，开始着手为即将在1995年发行的Netscape Navigator 2.0开发一个称之为LiveScript的脚本语言，当时的目的是同时在浏览器和服务器（本来要叫它LiveWire的）端使用它。Netscape与Sun公司联手及时完成LiveScript实现。就在Netscape Navigator 2.0即将正式发布前，Netscape将其更名为JavaScript，目的是为了利用Java这个因特网时髦词汇。Netspace的赌注最终得到回报，JavaScript从此变成了因特网的必备组件。

因为JavaScript 1.0如此成功，Netscape在Netscape Navigator 3.0中发布了1.1版。恰巧那个时候，微软决定进军浏览器，发布了IE 3.0 并搭载了一个JavaScript的克隆版，叫做JScript（这样命名是为了避免与Netscape潜在的许可纠纷）。微软步入Web浏览器领域的这重要一步虽然令其声名狼藉，但也成为JavaScript语言发展过程中的重要一步。

在微软进入后，有3种不同的JavaScript版本同时存在：Netscape Navigator 3.0中的JavaScript、IE中的JScript以及CEnvi中的ScriptEase。与C和其他编程语言不同的是，JavaScript并没有一个标准来统一其语法或特性，而这3种不同的版本恰恰突出了这个问题。随着业界担心的增加，这个语言标准化显然已经势在必行。

1997年，JavaScript 1.1作为一个草案提交给欧洲计算机制造商协会（ECMA）。第39技术委员会（TC39）被委派来“标准化一个通用、跨平台、中立于厂商的脚本语言的语法和语义”（http://www.ecma-international.org/memento/TC39.htm）。由来自Netscape、Sun、微软、Borland和其他一些对脚本编程感兴趣的公司的程序员组成的TC39锤炼出了ECMA-262，该标准定义了叫做ECMAScript的全新脚本语言。

在接下来的几年里，国际标准化组织及国际电工委员会（ISO/IEC）也采纳ECMAScript作为标准（ISO/IEC-16262）。从此，Web浏览器就开始努力（虽然有着不同程度的成功和失败）将ECMAScript作为JavaScript实现的基础。

尽管ECMAScript是一个重要的标准，但它并不是JavaScript唯一的部分，当然，也不是唯一被标准化的部分。实际上，一个完整的JavaScript实现是由以下3个不同部分组成的

- 核心（ECMAScript）——JavaScript的核心ECMAScript描述了该语言的语法和基本对象
- 文档对象模型（DOM）——DOM描述了处理网页内容的方法和接口
- 浏览器对象模型（BOM）——BOM描述了与浏览器进行交互的方法和接口

ECMAScript、DOM、BOM将是我们之后课程的主要内容。

JavaScript与Java

尽管名字中有Java，但是JavaScript和Java几乎没有什么共同点。Java是一种全功能的编程语言，是由Sun公司开发和推广的。Java是C和C++编程语言之后的又一种主流语言，程序员可以使用它创建完整的应用程序和控制消费电子设备。与其他语言不同，Java宣称具有跨平台兼容性；也就是说，程序员应该能够编写出可以在所有种类的机器上运行的Java程序，无论机器运行的是Windows、Mac OS X还是任何风格的UNIX。但实际上，Java不总是能够实现这个梦想，因为Sun和微软公司在这种语言的发展方向方面有很大的分歧。微软公司首先试图以自己的方式将Java集成到Windows中（Sun认为，这种方式使Java在Windows上以一种方式工作，而在其他机器上以另一种方式工作，从而破坏了Java的跨平台兼容性）；随后，微软公司从Windows中完全去除了Sun的Java，而创建了自己的类Java语言：C#。经过两公司之间的一轮诉讼之后，Sun占据了上风，现在可以在Windows（或Linux）上安装Sun的最新Java版本（http://www.java.com/getjava/）。Mac OS X操作系统在安装时会附带Java。

JavaScript可以做什么

用JavaScript可以做许多事情，使网页更具交互性，给站点的用户提供更好、更令人兴奋的体验。JavaScript使你可以创建活跃的用户界面，当用户在页面间导航时向他们提供反馈。例如，你可能在一些站点上见过在鼠标指针停留时突出显示的按钮。这是用JavaScript实现的，使用了一种称为翻转器（rollover）的技术 可以使用JavaScript确保用户在表单中输入有效的信息，这可以节省你的业务时间和开支。如果表单需要进行计算，那么可以在用户机器上的JavaScript中完成，而不需要任何服务器端处理。你应该知道一种区分程序的方式：在用户机

器上运行的程序称为客户端（client-side）程序；在服务器上运行的程序（包括后面要讨论的CGI）称为服务器端（server-side）程序。 可以使用JavaScript根据用户的操作即时创建定制的HTML页面。假设你正在运行一个旅行指南站点，用户点击夏威夷作为旅游目的地。你可以在一个新窗口中显示最新的夏威夷旅游指南。JavaScript可以控制浏览器，所以你可以打开新窗口、显示警告框以及在浏览器窗口的状态栏中显示定制的消息。JavaScript有一套日期和时间特性，可以生成时钟、日历和时间戳文档。 JavaScript还可以处理表单、设置cookie、即时构建HTML页面以及创建基于Web的应用程序。

JavaScript不能做什么

JavaScript是一种客户端（client-side）语言；也就是说，设计它的目的是在用户的机器上执行任务，而不是在服务器上。因此，JavaScript有一些固有的限制，这些限制主要出于安全原因：

- 1.JavaScript不允许读写客户机器上的文件。这是有好处的，因为你肯定不希望网页能够读取自己硬盘上的文件，或者能够将病毒写入硬盘，或者能够操作你计算机上的文件。唯一的例外是，JavaScript可以写到浏览器的cookie文件，但是也有一些限制
- 2.JavaScript不允许写服务器机器上的文件。尽管写服务器上的文件在许多方面是很方便的（比如存储页面点击数或用户填写的表单数据），但是JavaScript不允许这么做。相反，需要用服务器上的一个程序处理和存储这些数据。这个程序可以是用Perl或PHP等语言编写的CGI或Java程序。
- 3.JavaScript不能关闭不是由它自己打开的窗口。这是为了避免一个站点关闭其他任何站点的窗口，从而独占浏览器。
- 4.JavaScript不能从来自另一个服务器的已经打开的网页中读取信息。换句话说，网页不能读取已经打开的其他窗口中的信息，因此无法探察访问这个站点的冲浪者还在访问哪些其他站点。

我们的第一个脚本：最经典的HelloWorld程序!

```
<script type="text/javascript">
    document.write("<h2>Hello,JavaScriptWorld!</h2>");
</script>
```

开发环境

选择一个你喜欢的纯文本编辑器或<abbr title="集成开发环境">IDE</abbr>

NotePad++

VIM

UltraEdit

EditPlus

gEdit(Unix)

Emacs(Mac/Unix)

其它

至少一个符合W3C标准的浏览器（推荐火狐浏览器），和一些市场上流行的浏览器（IE）

FireFox 3.0+

Internet Explorer 6.0+ （由于IE具有多种不同的版本，还推荐安装IETester）

Google Chrome 1.0+

Opera 9.0+

Safari 3.0+

调试工具

FireFox下的FireBug,Venkman等

IE下的IE DeveloperToolbar,MS Script Debugger等(强烈不推荐MS Script Debugger,安装之后问题多)

Google Chrome 的JS控制台已经很强大了，Opera的错误控制台也可以，Opera蜻蜓和FireBug一样强大，Safari具有和Chrome一样的控制台

javascript快速入门2--变量,小学生数学与简单的交互

变量

对于变量的理解：变量是数据的代号。如同人的名字一样。

```
var num;//在JavaScript中使用关键字var声明一个变量
```

在JavaScript中,使用上面的语法,就可以声明一个变量,以便在之后给其指定值.

```
var num;
   num=128;//这样，就将num做为值128的一个名字，有了名字，就可以在之后引用！
   document.write(num);//输出128
```

貌似下面的代码和上面的具有一样的输出

```
   document.write(128);//这样当然也输出了128
```

不过再试试下面的代码

```
   document.write(3.14159265358979);//我们要多次输出这个值
   document.write(3.14159265358979);
   document.write(3.14159265358978);
   document.write(3.14159265358979);
   document.write(3.14159265358979);
```

于是可以用变量的概念来偷懒

```
var num;
   num=3.14159265358979;
   document.write(num);
   document.write(num);
   document.write(num);
   document.write(num);
   document.write(num);
```

虽然看上去没节约多少笔墨，但有一个概念即是：我们使用document.write多次输出的是同一个值！而前面一块则不同，它在每次输出时产生了一个新的值，显而易见，使用变量可以清楚的表达我们想要做的，而且简短的变量名引用可以使代码更清晰且不容易出错！ 当然，名字不是乱取的，变量的命名有一些限制：只能包含字母，数字，和下划线，还有个特殊的$字符，并且变量名只能以字母，下划线，还有$开头;另外，还不能使用JavaScript关键字和保留字;所以说，下面的变量声明全是错误的

```
var 34bad;//不能以数字开头
   var per人;//不能包含中文
   var bad-var;//非法的-
   var var;//var就是一个关键字，所以很明显不能用来做变量名
```

另外要注意的一点是,变量的名称是区分大小写的!

值，类型

```
var dog;
   dog="小虎子";//字符串，它们总被包含在双引号(或单号)中
   var num;
   num=1;//数字，它们裸露的出现了
   var strNum;
   strNum="1";//但是现在strNum所引用的是一个字符串,因为它被包含在引号中了
   var badNum;
   badNum=3.345;//一个小数，因为它带有一个小数点
   badNum=.2;//仍然是一个小数，这句代码与badNum=0.2是一样的!
   badNum = 0.4.5;//当然，这句代码是错的，一个非法数字
```

上面那样的写法（为了演示），我已经不想再忍受了，完全可以这样声明变量

```
var dog,num,strNum,badNum;//可以一次声明多个变量，它们用逗号分隔,然后再赋值
   dog="小虎子";
   num=1;
   ..........................
```

当然还有另一种声明变量的方法,事实上这种风格才是最常见的

```
var dog="小虎子"; var num=1; var str="some string",strNum="123";
   ......
```

数字（只能有整数或小数），字符串可能最常用的了，还有另一种类型：布尔值(Boolean).不像数字或字符串，有无限种可能的值，Boolean值只有两种可能：真，假

```
var bool=true;//用true表示真值
   bool=false;//用false表示假值
```

JavaScript是动态类型语言，在声明变量时无需指明其类型，在运行时刻变量的值可以有不同的类型。

```
var s="Hello,World!!!";//无需指明为字符串类型
   s=1.61803;//在运行时将变量值指定为另一个类型
```

JavaScript的变量类型不止字符串，数字，布尔值这三种，然而这三种确是最常用的了。其它数据类型（参考）：

复合（引用）数据类型是：

- 对象
- 数组

特殊数据类型是：

- Undefined

```
//事实上,我们接触的第一个数据类型是Undefined,它的含义是"未定义值"
        var a;//声明一个变量,但没有对其赋值
        alert(a);//但它仍然有值的,它的值为undefined
        alert(b);//但注意,输出一个未定义的变量将出现错误,而不是输出undefined
```

字符串

字符串相连

```
var s1="Hello,";
    s1=s1+"World!";
    alert(s1);
    s1+="!!!!";
    alert(s1);
```

数学运算与比较

首先是小学生都会的加减乘除:+,-,*,/

```
//加法 +
    //减法 -
    //乘法 *
    //除法 /
    //自增 ++
    //自减 --
    var a = 12; var b = 30; var c = a+b;
    alert(c);//输出42
    c=b-a;
    alert(c);//输出18
    c=c*2;
    alert(c);//输出36
    c=c/2;
    alert(c);//输出18
    c = 12;
    c++;//这与c=c+1;效果是一样的
    alert(c);//输出13
    c--;//这与c=c-1;效果是一样的
    alert(c);//输出11
    //自增与自减运算符出现的地方也有讲究
    c=20;
    alert(c++);//输出20,因为++写在变量后面,这表示变量c完成运算之后,再将其值增1
    alert(c);//现在将输出21,自减运算符也与些相似
    //如果只是类似这样的计算
    c = c+12; //可以这样写
    c+= 12;//这与写c= c+12;效果是一样的
    //类似其它的运算也有简便的方法
    c-=3;//c=c-3
    c*=4;//c=c*3;
    c/=2;//c=c/2;
```

要注意的是，在JavaScritp中，连接字符串时也使用“+”号。当字符串与数字相遇时？——JavaScript是弱类型语言

```
var num=23+45;
    alert("23+45等于"+num);//表达式从左往右计算，字符串之后的数字都会当成字符串然连接
    alert("23+45="+(23+45));//使用括号分隔
```

比较操作符:<,>,<=,>=,==,!=,!;比较操作符返回布尔值

```
//小于 <
    //大于 >
    //小于或等于 <=
    //大于或等于 >=
    //相等 ==
    //不相等 !=
    alert(2<4);//返回true
    alert(2>4);//返回false
    alert(2<=4);//返回true
    alert(2>=2);//返回true
    alert(2==2);//返回true
    alert(2!=2);//返回true
```

表达式的组合

```
    alert( (2<4)==(5>3)==(3<=3)==(2>=2)==(2!=2)==(2==2)==true );
```

逻辑运算符

逻辑运算符用于对布尔值进行比较

```
// &&逻辑与,当两边的值都为true时返回true,否则返回false
   // || 逻辑或,当两边值都为false时返回false,否则返回true
   // ! 逻辑非
   alert(true && false);//输出false
   alert(true && true);//输出true
   alert(true || false);//输出true
   alert(false || false);//输出false
   alert(!true);//输出false
```

类型转换入门

由于JavaScript是弱类型语言,所以我们完全可以将字符串和数字(两个不同类型的变量)进行相加,这个我们在前面已经演示过了.当然,不仅仅可以将字符串和数字相加,还可以将字符串与数字相乘而不会出现脚本错误!

```
var str ="some string here!"; var num = 123;
   alert(str*num);//将输出NaN,因为乘法运算符只能针对数字,所以进行运算时计算机会将字段串转换成数字
   //而这里的字符串转换成数字将会是NaN
   //NaN是一个特殊的值,含义是"Not A Number"-不是一个数字,当将其它值转换成数字失败时会得到这个值
   str ="2";
   alert(str*num);//将输出246,因为str可以解析成数字2
```

其它类型转换

```
var bool = true;
   alert(bool*1);//输出1 ,布尔值true转换成数字为1,事实上将其它值转换在数字最简单的方法就是将其乘以1
   bool = false;
   alert(bool*1);//输出0
   alert(bool+"");//输出"flase",将其它类型转换成字符串的最简单的方法就是将其写一个空字符串相连
   alert(123+"");//数字总能转换成字符串
   var str = "some string";
   alert(!!str);//true,因为非运算符是针对布尔值进行运算的,所以将其它类型转换成布尔值只须将其连续非两
   str ="";
   alert(!!str);//输出false,只有空字符串转换成布尔值时会是false,非空字符串转换成布尔值都会返回true
   var num =0;
   alert(!!num);//false
   num=-123.345;
   alert(!!num);//true,除0以外的任何数字转换成布尔值都会是true
   //还有一个非常重要的是,空字符串转换成数字将会是0
   alert(""*1);//输出0
```

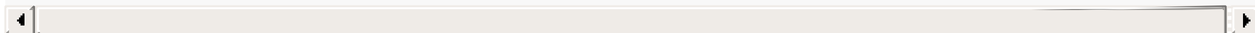
获取变量类型 typeof 运算符

```
var bool = true;
   alert(typeof bool);//输出boolean
   var num =123;
   alert(typeof num);//输出number
   var str = "some string here";
   alert(typeof str);//输出string
   var strNum = "123";
   alert(typeof strNum);//输出string
   strNum *= 1;
   alert(typeof strNum);//输出number
```

根据用户的输入进行计算

使用window.prompt可以弹出包含输入框的窗口，用户输入的内容将是此函数的返回值

```
var msg = window.prompt("请输入一个数字","默认值");//prompt方法包含一个必须参数和一个可选参数。
   //第一个参数是必须的，将在窗口中显示的文字，第二个参数是可选的，为文本框的预设值
   alert(msg);//将输出我们输入进去的东西
```



请用户输入年龄，我们计算出他活了多少秒（貌似无聊的很啊）

```
var age=window.prompt("请输入您的年龄！","18"); var liveSeconds=age*365*24*3600;
   alert("您已经度过了"＋liveSeconds＋"秒!");
```

javascript快速入门3--分支判断与循环

分支结构

- 单一选择结构（if）
- 二路选择结构（if/else）
- 内联三元运算符 ?:
- 多路选择结构（switch）

```
var condition = true; if (condition) {
        alert("我将出现！");
    }
    condition = false; if (condition) {
        alert("我不会出现！");
    } else {
        alert("我会出现！");
    }
    condition ="some string"; if (condition) {
        alert("可以直接对任何数据类型进行if判断,在判断时计算会自动将其转换成布尔值!");
    } var val = condition?"当为true时我将被返回":"当为false时我将被返回";
    alert(val);//将输出"当为true时我将被返回"
```

对于if..else语句，如果要执行的语句只有一条，可以不使用“｛｝”，但这种写法并不推荐.但确实这样可以简化代码:

```
var str ="one"; if (str=="one") alert("str的值为字符串'one' !"); else alert("not one");
```

虽然JavaScript中没有if .... elseif 结构,但可以使用if...else的简写方式得到

```
//为了判断用户输入的成绩的范围,我们使用了多重嵌套的if .. else语句
   var num = window.prompt("请输入XXX的成绩!","");
   num *=1;//window.prompt方法始终只返回字符串,用这样的方法将其转换成数字
   if (isNaN(num) && num) {//不能将其它值转换成数字时将返回NaN,可以用内置的isNaN判断值是否是NaN
       alert("您输入的不是一个数字!");
   } else { if (num<=100 && num>=90) {
           alert("Excellent!");
       } else { if (num =80) {
               alert("Good!");
           } else { if (num < 80 && num >= 70) {
                   alert("So so!");
               } else { if (num < 70 && num >=60) {
                       alert("Be careful !!!");
                   } else { if (num < 60 && num >= 0) {
                           alert("Oh, NO!");
                       } else {
                           alert("USB!");
                       }
                   }
               }
           }
       }
   } //上面的代码由于用了多重的if..else嵌套,显得非常的混乱,简化只须将else后的花括号去掉就行了
   // if (...) {...}这算是一句
   if (isNaN(num)) {
       alert("您输入的不是一个数字!");
   } else if (num<=100 && num>=90) {
       alert("Excellent!");
   } else if (num =80) {
       alert("Good!");
   } else if (num < 80 && num >= 70) {
       alert("So so!");
   } else if (num < 70 && num >=60) {
       alert("Be careful !!!");
   } else if (num < 60 && num >= 0) {
       alert("Oh, NO!");
   } else {
       alert("USB!");
   } //看上去清晰多了,但要注意的是,JavaScript中没有elseif 这样的语法,所以上的else if之间是有空格的
```

用于进行多次判断的switch语句

```
switch(condition) { //switch本来就是跳转的意思（又称为“开关”），所以switch语句就是判断情况，跳到符
       case 4:
           alert("c的值是4"); case 3:
           alert("c的值肯定大于等于3"); case 2:
           alert("c的值肯定大于等于2"); case 1:
           alert("c的值肯定大于等于1");
   } //可以使用 break来只执行符合一个条件的语句
   switch(condition) { case 4:
           alert("c的值是4"); break; case 3:
           alert("c的值是3"); break; case 2:
           alert("c的值是2"); break; case 1:
           alert("c的值是1"); break;
   } var condition="one"; switch(condition) {//switch不但可以用来判断数字,还可以判断字符串，甚
       case "one":
           alert("condition的值是字符串'one' !"); break; case "three":
           alert("condition的值是字符串'three' !"); break; case "four":
           alert("condition的值是字符串'four' !"); break; case "five":
           alert("condition的值是字符串'five' !"); break; default://当所有情况都不匹配时，将执
           alert("我们要万无一失!condition什么都不是!");
   }
```

循环

循环用来指明当某些条件保持为眞时要重复的动作。当条件得到满足时，就跳出循环语句。在 JavScript 中有四种循环结构可用。

- 由计数器控制的循环（for）
- 在循环的开头测试表达式（while）
- 在循环的末尾测试表达式（do/while）
- 对对象的每个属性都进行操作（for/in）

for 语句指定了一个计数器变量，一个测试条件，以及更新该计数器的操作。在每次循环的重复之前，都将测试该条件。如果测试成功，将运行循环中的代码。如果测试不成功，不运行循环中的代码，程序继续运行紧跟在循环后的第一行代码。在执行该循环后，计算机变量将在下一次循环之前被更新。

```
for (var i=0;i<10;i++) {//for循环的圆括号里面须放三个句子,分别是1.初使化计数器 2.判断条件 3.更新计
        alert("i当前的值为"+i);
    }
```

其实for循环语句完全可以这样写,下面的代码和上面的效果是一样的（虽然没有必要，但从这样的代码可清楚看出for循环如何工作的）

```
var i=0;//循环进行之前初始化i
    for(;;) {//for语句括号中必须有三个语句，但可以为空语句
        if (i<10) {//当条件为true时才执行代码
            alert("i当前的值为"+i);
        } else {//当条件为false时就退出循环
            break;//使用break退出循环
}
    }
```

一个死循环最能说明while的工作方式了（但这样的错误我们绝不能在实际编程中出现）

```
while (true) {
        alert("你关不掉我的！");//这就是网上那些所谓的高手写的“关不上的窗(周传雄新歌,力荐)”代码
    }
```

do..while循环与while循环不同之处在于它至少将代码块中的代码执行一次

```
do {
        alert("我肯定会出现一次的");
    } while (false);
```

而对于for ... in循环，我们将在讲解数组和对象时使用

javascript快速入门4--函数与内置对象

函数

函数(又称为方法)用于对一大段为了达到某种目的的代码进行归类，以使代码更具有条理：

```
        //一段计算三角形面积的代码
    var wide=window.prompt("请输入三角形的底边长度！",""); var high=window.prompt("请输入三角形
    if (isNaN(area)) {//判断用户是否输入的是数字
        alert("三角形的面积为"+area);
    } else {
        alert("您的输入有误!");
    }
```

如果我们需要在其它地方使用此功能，那么最简单的方法就是Ctrl＋C然后Ctrl＋V，使用函数可以节约一些代码

```
function calcAngleArea() { //使用function关键字声明一个函数，接着是函数的名称，函数名称必须符合变量
        //花括号用来表示一段代码块
        var wide=window.prompt("请输入三角形的底边长度！",""); var high=window.prompt("请输入三
        if (isNaN(area)) {
            alert("您的输入有误!");
        } else {
            alert("三角形的面积为"+area);
        }
    }
```

但是写了这样了个计算三角形面积的函数之后，页面打开时并没有任何东西会出现，那是因为函数必须使用“函数名（）”这样的语句来调用执行，所以我们还需要再添加一句calcAngleArea();如果要多次进行这样的计算，只需多次调用些函数即可!(事实上，看到这样的格式，我们发现，像alert(),prompt(),isNaN()这些也是函数，它们是系统内置的函数!)

```
    //在之前我们已经声明了计算三角形面积的函数calcAngleArea
calcAngleArea();
    calcAngleArea();
    calcAngleArea(); //将会进行三次这样的计算
```

当然，我们可以对其进行一些改进，以使这个函数更好用

```
function calcAngleArea(wide,high) {//具有参数的函数，参数其实是一些变量，多个参数用“，”分隔
        var area=wide*high/2;
        if (isNaN(area)) {
            alert("您的输入有误!");
        } else {
            alert("三角形的面积为"+area);
        }
    }
```

这样，函数就具有伸缩性了，我们不必强制用户在prompt弹窗中输入内容。我们先测试一下函数如何执行

```
calcAngleArea(12,8);//传入参数12和8，在函数内部执行。接着我们就看到了输出
```

同样，有时候我们并不是相让用户输入某些值，也并不想将某些值输给用户。但现在这个计算三角形面积的函数不管我们想如何处理结果，它都只是在弹窗中将结果显示给用户。这个时候就用到了函数返回值的功能：

```
  //在函数内部可以使用return语句将值返回给调用函数的上下文
     function calcAngleArea(wide,high) { var area=wide*high/2;
         if (isNaN(area) || !area) { return false;
             alert("注意,一个函数中的return执行之后，函数就停止运行了，所以你不会看到我!");
         } else { return area;
         }
     } var a=calcAngleArea(23,8);//执行函数，函数的返回值将会赋给变量a
     if (a) {
         document.write(a);//当函数有返回值时，我们可以以想要的任何方式来输出a,而不是预先定义好的aler
     }
```

这样，这个函数的功能就是真正的无瑕眦的计算三角形面积的函数了（尽管看上去有些简单），我们输入宽和高，然后函数将其计算后将结果返回，如果只是像"calcAngleArea(23,8)"这样调用函数的话,返回的结果将会丢失，所以我们用一个变量将结果保存了下来。

函数所带来的作用域问题

在函数内部声明的变量（局部变量），在函数外部并不能访问

```
     function demo() { var a="外面不能访问我！";
     } //alert(a);//出错，没有声明变量a
     demo();//执行函数
     alert(a);//仍然出错
```

但在函数外部声明的变量（全局变量），在函数内部是可以访问的

```
    function demo() { //在一个脚本中，使用function关键字声明的具有名称的函数在脚本中出现的次序是任意的
  alert(globalVar);
     } //demo();//出错，执行时变量globalVar还没有声明
     var globalVar="Hello!";
     demo();//输出Hello!
```

内置对象Math与Date

Math对象为我们提供了很多用于数学计算的方法和一些常量

```
 alert(Math.PI);//输出π
alert(Math.pow(10,3));//输出10的3次方
alert(Math.abs(-12));//输出-12的绝对值
var num=23.34;
alert(Math.ceil(num));//返回大于等于num的最小整数
alert(Math.floor(num));//返回小于等于num的最大整数。
alert(Math.round(num));//返回与num最接近的整数(四舍五入)。
alert(Math.random());//返回介于 0 和 1 之间的伪随机数。产生的伪随机数介于 0 和 1 之间（含 0，不
alert(Math.max(2,3,4));//返回多个数值参数中较大的那个
alert(Math.min(2,3,1));//返回多个数值参数中较小的那个
alert(Math.sqrt(2));//返回一个数的平方根
alert(Math.SQRT2);//返回2的平方根
alert(Math.SQRT1_2);//返回二分之一的平方根
```

Date对象像一个时光机

```
var d = new Date();//Date对象需要创建
   //Date 对象能够表示的日期范围约等于 1970 年 1 月 1 日前后各 285,616 年。
   alert(d);//直接输出这个对象,将会得到一个表示时间的字符串
   //这个对象有一些方法,可以用来获取时间的各个部分
   alert(d.getYear());//获取年,2000年以前返回年份后两位,2000年之后的返回年份的完整表达方式
   alert(d.getFullYear());//始终返回年份的4位数表达方式
   alert(d.getMonth());//返回月份,注意,月份是从0开始计数的,所以1月时将返回0
   alert(d.getDate());//返回今天几号
   alert(d.getDay());//返回今天星期几,星期天是0,星期1是1....
   alert(d.getHours());//返回小时
   alert(d.getMinutes());//返回分钟
   alert(d.getSeconds());//返回秒
   alert(d.getMilliSeconds());//返回毫秒
alert(d.getTime()); //返回一个整数值,这个整数代表了从1970年1月1日开始计算到Date对象中的时间之间的毫
   //日期的范围大约是1970年1月1日午夜的前后各285616年,负数代表1970年之前的日期
```

我们不但能从中获取时间值,还可以设置时间值

```
var d = new Date();
   d.setFullYear(1990);//设置年份为1900
   alert(d.getFullYear());//返回1900
   alert(d.getTime());//输出的值是负的,这验证了上面所说的getTime()返回值
```

与那些getXXX方法对应的设置时间的函数仅仅是将get改成set

```
var d=new Date();
   d.setFullYear(2004);
   d.setMonth(11);//设置月份为12月,注意月份是从0开始计数的
   d.setDate(2);
   d.setHours(6);
   d.setMinutes(12);
   d.setSeconds(12);
   alert(d.getDay());//输出2004年12月2日星期几
   //setXXX这样的方法有个最大的好处就是如果我们设置了错误的值,脚本并不会出错,但日期会自动更正
   d= new Date();
   d.setYear(2003);
   d.setMonth(1);//月份从0开始计数
   d.setDate(31);//2月从来不会有31号
   alert(d);//输出日期,发现会是3月3号
```

运用Date对象这个自动更正的好处,我们可以用它来判断用户输入的日期是否是有效的

```
//让用户输入生日
    var year=window.prompt("请输入出生年份!","")*1; var month=window.prompt("请输入出生月份!",
        alert("您的输入有误!");
    } else { var timeMachine=new Date();
        timeMachine.setFullYear(year);
        timeMachine.setMonth(month-1);//记住,月份是从0开始计数的
timeMachine.setDate(date); var trueYear = timeMachine.getFullYear(); var trueMonth = tim
        var trueDate = timeMachine.getDate(); if (trueYear != year || trueMonth != month
            alert("您撒谎!");
        } else {
        alert("虽然这个时光机能知道输入的日期是否有效,至于人是不是在那天生的,它是不能去看一看的!");
        }
    }
```



javascript快速入门5--数组与对象

数组

数组,实际上就是将一大堆相似的数据有秩序的放在格子箱中,十分像药房里的那些柜子.

```
| 数据1 | 数据2 | 数据3 | 数据4 | 数据5 | 数据6 |
```

用代码创建数组

```
var arr = new Array();//Array和Date一样,也是一个内置对象,需要使用new运算符创建
   arr[0]="数据1";//向数组中添加一个元素,数组中的元素是有编号的,并且要注意的是,编号从0开始
   //上面一行代码就向数组中的第一个箱添加了一个元素
   arr[1]="数据2";//方括号用以指定下标1
   arr[2]="数据3";
   arr[3]="数据4";
   arr[5]="数据6";
   arr[4]="数据5";
   alert(arr);//将会输出"数据1,数据2,数据3,数据4,数据5,数据6"   是以逗号分隔的字符串
   //并且这些字符串的连接是按(下标)顺序的
   alert(arr[0]);//当然,我们也可以直接访问其中第一个元素
   alert(arr[1]);//第二个
   alert(arr.length);//遇到的第一个数组对象的属性,length属性用以表示数组中元素的个数,输出6
```

遍历数组 for 循环

```
for (var i=0;i< arr.length;i++) {
       arr[i]+="  ---changed";//将数组中每个元素(字符串)后面连上一个"  ---changed"
}
   alert(arr);//变了
```

创建数组的其它方式

```
var arr;
   arr = new Array();//这样创建了一个空数组
   alert(arr);//输出为空,因为没有元素
   arr = new Array(3);//在申明时只放一个正整数表示数组的长度
   alert(arr.length);//输出3
   alert(arr);//输出两个逗号,它里面放了3个空元素
   //申明时指定了数组的长度,然后修改指定位置的值
   arr[2]="end";//将最后一位修改为"end"
alert(arr); //并不是在数组申明时指定了长度就不能更改,更改长度很简单
   arr[8]="super";//数组长度将自动增长到8
   //记住,JavaScript中数组长度会自动增长,并且length属性会自动更新
   alert(arr.length);//输出9,JavaScript中数组下标是从0开始的
   alert(arr[8]); //也可以在创建数组时就指定值
   arr = new Array(1,2,3,4,5,6,7,8);
   alert(arr); //如果在创建数组时只指定一个值,并且是正整数
   arr = new Array(6);//将得不到预期效果,因为这是在声明一个长度为6的空数组
   //需要这样
   arr = new Array();
   arr[0]=6; //还可以使用数组字面量的方式
   arr = [];//是的,一个空中括号
   //与下面一句几乎是等价的
   arr = new Array(); //但更灵活和简便
   arr =[3];//将创建一个长度为1,第一个元素为3的数组
   arr = [2,3,4,6];//多个元素以逗号分隔
   alert(arr[0]);//输出2,下标的顺序与在中括号中出现的顺序相关
   //数组中可以混合存放字符串,数值,布尔值...,几乎所以类型的值,包括数组
   arr = new Array(1,0,true,"some string",new Array("a",3));//第五个元素放的是一个数组
   alert(arr[4]);//输出"a",3
   alert(arr[4][0]);//输出"a"
```

数组的按引用传值的特性

```
var arr = [2,3,4]    ; var arr2 =arr;//这相当于给arr取了个别名
   arr2[0]=234;
   alert(arr[0]);//输出234,因为arr与arr2是同一个对象
```

向数组中添加,删除元素(push,delete)

```
var arr = [2,4];
   arr.push(6);//push方法将元素添加到数组末尾
   alert(arr.length);//输出3
   arr.push("a","b");//可以一次添加多个元素
   alert(arr.length);//输出5
   alert(arr[5]);//输出"b"
   alert(arr.push(123));//push方法执行后会返回数组的新长度值,输出6
   //事实上将元素添加到数组末尾的最简单的方法是
   arr = [4,5];
   arr[arr.length]="new element";//利用数组长度自动增长的特性
   alert(arr.length);//输出3
   //为了更明了的明白push的工作原理,我们可以使用一个简单的自定义函数来完成这项工作
   function array_push(element,arr) {//第一个参数为要添加的元素,第二个参数为该数组
       arr[arr.length]=element; return arr.length;
   }
   arr = [1,2,3];
   array_push(345,arr);
   alert(arr.length);//输出4
   alert(array_push("some string",arr));//输出5
alert(arr); //删除一个元素
   arr = ["#","$","%"];
   alert(arr); delete arr[2];
   alert(arr);
   alert(arr.length);//元素被删除了,但数组长度并没有变化,因为这样能使我们使用相同的下标访问以前的元素
   //使用delete与下面的语句效果一样
   arr = ["#","$","%"];
   alert(arr);
   arr[2]=undefined;//undefined是一个值
   alert(arr);
```

join方法,返回数组中的所有元素以指的分隔符间隔的字符串

```
var arr = [2,3,4];
   alert(arr.join("#")); //事实上当我们直接输出数组时,系统会自动调用这样的方法
alert(arr);
   alert(arr.join(","));//两句的输出效果是一样的
```

对象

对象是一堆属性的集合,其实它和数组是相通的

```
var obj = new Object();//创建一个对象
   obj.property = "value";//使用点语法给对象添加属性
   alert(obj);//只会输出含糊的[object Object]
   alert(obj.property);//真正的数据全存储在它的属性上面
```

对象吗?就当和现实中的对象一样:一个"人"对象

```
var person = new Object();
   person.age = 18;
   person.weight = "123kg";
   person.height = "170cm";
   person.arm = 2;//两个臂膀
   person.leg = 2;
```

上面创建的对象,描述了现实中的人的一些特性:年龄 18;重量 123kg;身高 170cm;不是残疾(这个是我推断的); 其实数组也能完成这样的工作

```
var person = new Array();
   person[0] = 18;
   person[1] = "123kg";
   person[2] = "170cm";
   person[3] = 2;
   person[4] = 2;
```

但是这样的表达方式,没人能看出这是一个"人"对象,使用数字下标没有对象的属性明了,代码难于理解. 其实数组可以使用字符串下标的

```
var person = new Array();
   person["age"] = 18;//注意,中括号里的下标是一个字符串,所以需要使用引号
   person["weight"] = "123kg";
   person["height"] = "170cm";
   person["arm"] = 2;
   person["leg"] = 2;
```

我说过了,数组和对象是相通的

```
var arr = new Array(); var obj = new Object();
   alert(typeof arr);//object
   alert(typeof obj);//object
```

所以,数组用字符串下标,事实上也是在给其添加属性

```
var arr =[1,2,3];
   arr["property"]="some data";
   alert(arr.property);//"some data"
   //但注意的是,数组的length属性只能报告具有数字下标的元素的个数
   alert(arr.length);//3
```

而对象也可以使用类似语法访问它的属性

```
var obj = new Object();
   obj.property = "some data";
   alert(obj["property"]);//"some data"
   //当然也可以使用数字下标
   obj[1]=123;
   alert(obj[1]);//123
   alert(obj.property);//"some data"
   alert(obj.length);//但与数组不同的是,它没有length属性
```

与数组字面量相对应的,也有对象字面量的声明方式

```
var obj = {
       a:123,//这里的a,b等同样是对象的属性名
       b:456 //注意,最后没有逗号
};
   alert(obj.a);
   alert(obj.b); //还可以这样写
   obj = { "a":345,  //虽然如果用引号引起来就可以使用空格等不符合变量命名规范的字符,但强烈不推荐
       "b":333 };
   alert(obj.a);//345
```

对于数组,我们可以使用for对其进行遍历,但for循环只能遍历具有数字下标的元素

```
var arr =[1,2,3,4];
   arr["stringIndex"]="some data";//这个不会被遍历到
   alert(arr.length);//arr.length属性也不报告包含此元素
   for (var i=0;i< arr.length;i++) {
       alert(arr[i]);//i只会是数字,所以不能遍历字符串下标的元素
   }
```

我们之前看到,对数组使用字符串下标实际上是给这个数组对象添加属性,这个时候我们会发现,对象的属性还没有什么好的方法列举出来,for.. in...循环出现了 (对于研究对象,for in循环太有用了)

```
var obj={
       age:12,
       height:170 }; for (var i in obj) {//i将会被列举为键名,就是所说的字符串的下标
       alert(i+"\n"+obj[i]);/*将会以类似
           age
           12
       这样的格式分别输出它的键名键值对 */ }
```

for in 循环不但是用来遍历对象属性,它也可以遍历出数组的具有数字下标的元素

```
var arr = [1,2,3,4,5,6];
   arr["property"]=78;//会被遍历到,因为它是属性
   for (var i in arr) {
       alert(i+" : "+arr[i]);
   }
```

了解这些之后,我们可以使用它们来存储一些数据:一个班的学生的成绩(该用数组还是对象呢?这确实是一个问题)

```
//该是对象就是对象,该是数组就是数组
   var myClass=[];//创建一个数组,放置每个学生的信息,以学生的学号作为数组下标
   myClass[1]={ "name":"HUXP", "Chinese":60, "English":70, "Math":80, "Grade":"C" };
   myClass[2]={ "name":"发哥", "Chinese":80, "English":80, "Math":80, "Grade":"B" };
   myClass[3]={ "name":"Per", "Chinese":66, "English":77, "Math":88, "Grade":"B" };
   myClass[4]={ "name":"小虎子", "Chinese":60, "English":60, "Math":770, "Grade":"C" };
   myClass[5]={ "name":"DBD", "Chinese":70, "English":70, "Math":70, "Grade":"B" };
   myClass[6]={ "name":"○", "Chinese":77, "English":77, "Math":80, "Grade":"B" };
   myClass[7]={ "name":"Me", "Chinese":100, "English":100, "Math":100, "Grade":"A", "Say
   alert(myClass[5].Chinese)//如果有学号,输出对应学号的学生的语文成绩太简单了
   //更复杂的,搜索学生姓名,返回他的所有信息,是使用函数的时候了
   function searchByName(name) { for (var i=1;i< myClass.length;i++) { if (myClass[i].na
           }
       }
   }
   alert(searchByName("○").Math);
```

String对象以及一些用于字符串的方法和属性

创建String对象

```
var str = new String();
   alert(str);//输出空字符串
   str = new String("some string here");
   alert(str);//输出字符串"some string here"
   //表面上看,这和直接创建的字符串是一样的效果
   str = "some string here";
   alert(str);
```

但由于使用new String();所以创建出来的是对象

```
var str = new String();
   alert(typeof str);//object
   //因为是对象,所以自然有很多属性和方法
   //但字符串本身也存这样的方法
```

有很多用于处理字符串的方法以及一些属性

- length 属性,返回字符串的长度
- indexOf 方法,返回字符串内第一次出现子字符串的字符位置
- lastIndexOf 方法,返回字符串中子字符串最后出现的位置
- charCodeAt 方法,返回一个整数，代表指定位置上字符的 Unicode 编码
- fromCharCode 方法,从一些 Unicode 字符值中返回一个字符串
- replace 方法,进行文字替换,返回替换后的字符串的复制
- substr 方法,返回一个从指定位置开始的指定长度的子字符串
- substring 方法,返回位于 String 对象中指定位置的子字符串
- toLowerCase 方法,返回一个字符串，该字符串中的字母被转换为小写字母
- toUpperCase 方法,返回一个字符串，该字符串中的所有字母都被转化为大写字母
- split 方法,把字符串分割为字符串数组。

```
var str ="some string here";
   alert(str.length);//16
   alert(str.indexOf("s"));//0,字符串的位置从0开始计数
   alert(str.indexOf("o"));//1
   alert(str.indexOf("k"));//没有找到时返回-1
   alert(str.lastIndexOf("e"));//15,从后往前查找
   alert(str.charCodeAt(0));//115,小写s的Unicode编码
   alert(String.fromCharCode(65,66,67,68));//返回ABCD,注意fromCharCode是String对象的静态方法
   alert(str.replace("some","much"));//"much string here"
   alert(str.substr(1,2));//uc,从下标1开始向后截取2个字符
   alert(str.substring(1,2));//c,从下标1开始截取到下标2,不包括结束位置的字符
alert(str.toLowerCase());
   alert(str.toUpperCase());
   alert(str.split(" "));//some,string,here
```

javascript快速入门6--Script标签与访问HTML页面

Script标签

script标签用于在HTML页面中嵌入一些可执的脚本

```
<script>
        //some script goes here
    </script>
```

script标签有三个特殊的属性(当然,像id,class这样的属性它也是有的,HTML页面中几乎每个元素都可以有class,id属性)

```
<script language="JavaScript">//language属性指明标签里包含的脚本所使用的语言
        //它有三个常见的取值JavaScript,JScript,VBScript
        //some script goes here
    </script> //对于JScript只有IE能够识别,其它浏览器会忽略这个标签其里面的内容
    //而对于VBScript,只有Windows上的IE能够识别,运行
    //然而language属性后来在XHTML中被type属性替代了 <script type="text/javascript">//取值也变了
        //some script goes here
    </script>
```

在Web浏览器中,我们只会使用JavaScript,type属性设置为text/javascript.事实上,由于JavaScript十分流行,它几乎成了脚本的代名词,而在Web浏览器中,即使script标签不加任何属性,浏览器也会把它当成JavaScript

```
<script> alert("Hello!"); </script> //上面的那段代码将会按JavaScript的方式运行
    //即使有IE中,不加声明的script块也会当成JavaScript执行,而不是VBScript <script> msgbox "Hello
    </script> //上面的代码在IE中也会报错,IE也会将其当成JavaScript执行
```

以前在HTML页面中,一些标签常会加一些诸如onclick,onmouseover这样的属性,这是一种事件绑定(关于事件,我们之后会详细讲解的,不要急).用于指定当HTML页面某个元素上发生了什么事的时候去执行的JavaScript代码(当然也可以是其它客户端脚本)

```
<img src="../images/stack_heap.jpg" alt="内存堆栈" onclick="alert('你把我点疼了!')"  />
```

上面的代码将在HTML页面上显示一个图像,当你用鼠标点击一下时,会出现一个弹窗,显示'你把我点疼了!',onclick属性的值其实是一段JavaScript代码;这就是事件,下面是其它一些简单的事件

- onclick ,当鼠标点击一下时执行一次
- onmouseover ,当鼠标放上去时执行一次

- onmouseout ,当鼠标移出去时执行一次
- onmousedown ,当鼠标按下时执行一次
- onmouseup ,当鼠标在上面松开(弹起)时执行一次
- onmousedblclick ,当鼠标双击时执行一次
- onload ,当对象加载完成时执行一次

以前网上十分流行的称之为RollverImages(翻转图像)的效果其实就是组合onmouseover,onmouseout这样的事件和简单的JavaScript代码实现的

```
<img src="../images/stack_heap.jpg" alt="内存堆栈" onmouseover="this.src='../images/over.j
```

你可能知道,onmouseover这类的属性中的字符串将会在事件发生时当成脚本来执行,但上面的那些代码看上去十分模糊

```
//为了便于查看,我们将它们提取出来放在下面
this.src='../images/over.jpg'
this.src='../images/out.jpg'
```

分析上面的代码,我们发现,这其实是在给一个对象this的属性src赋值,但问题是我们并没有声明过一个叫this的对象!其实this对象是一个一直存在的一个对象,它不能被声明(this是关键字).这里,this就是指"这个",指这个标签!对于HTML中的元素,JavaScript会自动将其解析成一个对象.对于下面的img标签,会解析成下面一个对象:

```
<img src="../images/stack_heap.jpg" alt="内存堆栈" onclick="alert('Hello!')"  /> //注
this = {
    src:"../images/stack_heap.jpg",
    alt:"内存堆栈",
    onclick:"alert('Hello!')",
    tagName:"IMG"
};
//其实不止这些属性
```

上面,img标签会被解析成一个对象,具有src,alt等属性,src,alt属性是我们写在HTML里面的,而tagName则是系统自动生成的,它表示标签的标签名!我们可以用下面的代码进行测试:

```
<img src="../images/stack_heap.jpg" alt="内存堆栈" onclick="alert(this.src);alert(this.ta
```

自然,我们也可以修改它的值,于是翻转图象的效果就这样成功了

对于这样的行内事件绑定,有一些注意点.

```
<head>
        <script>
           function myFn() {
               alert("图象加载完成了!");
           } </script>
   </head> //............若干若干代码之后 <img src="../images/stack_heap.jpg" alt="内存堆栈
```

上面的代码执行是没问题的,然而将顺序翻转一下(script可以放在任何合法的地方)

```
<img src="../images/stack_heap.jpg" alt="内存堆栈" onload="myFn()"  />//当图象加载成功时执行-
   //............若干若干代码之后 <script>
       function myFn() {
           alert("图象加载完成了!");
       } </script>
```

HTML页面按照从上往下的顺序加载执行,当图象加载成功后,就去执行onload里的内容(一个自定义函数),但由于script标签在下面若干代码之后,所以还没被加载,因此会出错"myFn is undefined";这就是为什么要将script标签放在head部分的原因,因为head在body前面,当body里的元素加载完成时,head中的script肯定加载完成了

但后来有了XHTML,有了"三层分离",W3C推出了DOM2,我们需要使用另一种方式绑定事件,获取HTML页面元素.再以上面的图像为例:

```
<head>
        <script>
            var img = document.getElementById("myImg");//我们通过ID来获取它
            //document.getElementById方法有个参数,一个字符串ID
            //然后,img就表示了那个图像标签对象
img.onclick = myFn; /*我们不是把JavaScript代码以字符串符值给它的onclick属性
            而是直接赋值给它一个函数名
            你也会说,为什么不是img.onclick=myFn();
            因为现在是在JavaScript代码区域中
            加"()"表示执行这个函数,然后将这个函数的返回值赋给了img.onclick*/
           function myFn() {
               alert("图象加载完成了!");
           } </script>
   </head> //....... <img src="../images/stack_heap.jpg" id="myImg" alt="内存堆栈" /> //我
```

但上面的代码执行了仍会出错,因为HTML从上往下加载,当加载到script时,body部分在下面,还没有被加载,而JavaScript在浏览加载到时就会自动执行.这时,页面上的ID为myImg的img还没被加载到,所以会出错;document.getElementById方法需要一个字符串形式的ID,而如果页面上没有ID为这个的元素,它则会返回null(空对象);而在下面一行,img.onclick这一句使用了一个空对象,所以会在这里出错!至于解决方法,其实只是将传统的行内事件绑定的script位置反过来放.将script放在所以HTML元素后面!

```
<img src="../images/stack_heap.jpg" id="myImg" alt="内存堆栈" /> //..................若干作
        var img = document.getElementById("myImg"); //这个时候,由于script标签放置的位置处在im
img.onclick = myFn; function myFn() {
           alert("图象加载完成了!");
       } </script>
```

但标准仍然推荐将script放在head部分!那么,这就要用到另一个事件onload

```
window.onload = initAll;//将所有代码写在一个函数之中,然后注册到window对象的onload事件属性上
//window表示窗口对象,只要窗口打开,它就始终存在,当页面加载完成后,会触发window对象上的onload事件
function initAll() { var img = document.getElementById("myImg");
     img.onclick = myFn; function myFn() {
        alert("图象加载完成了!");
    }
}
```

这样,代码就不出错了,不管将脚本放在什么位置,initAll只有当页面加载完成后才会被执行

访问HTML页面:<abbr title="HTML 文档对象模型">HTML DOM</abbr>

HTML DOM将整个页面当成一个document对象,HTML里的标签都要通过document对象来访问.而文档中的每个标签,又会转换成一个对象

```
<p class="demo" title="第一个段落:DOM树" id="p1">我们用一个p标签来演示</p>
```

它又会被转换成下面这个对象

```
//总该记得对象字面量语法吧
{
       tagName:"p",
       className:"demo",
       title:"第一个段落:DOM树",
       id:"p1",
       innerHTML:"我们用一个p标签来演示" } //你也许会奇怪,为什么标签的class属性会变成对象的classNa
   //class是JavaScript保留字!!!
   //tagName表示它的标签名,而innerHTML表示它里面的HTML代码
```

浏览将HTML标签转换成这样的对象后,在JavaScript中访问该标签的属性或里面的内容就简单多了,但问题是如何访问到这个对象!!

```
//首先要给该标签加个ID,然后使用document.getElementById方法就能够访问到它了
   window.onload = initAll;//注意,要将所要访问HTML页面的代码都放在window的onload事件处理上!
   function initAll() { var p = document.getElementById("p1");
       alert(p.className);
       alert(p.tagName);
       alert(p.title);
       alert(p.id);
       alert(p.innerHTML);
   }
```

访问HTML页面就这么简单!获取一个元素之后,不但可以读取它的属性值,还可以设置它的属性值!

```
   window.onload = initAll; function initAll() { var p = document.getElementById("p1");
       p.title="JavaScript";
       p.className="load";//我们可以更改它的样式
   }
```

利用这些,我们已经能做出一些激动人心的事了!

```
//一些CSS
   .over { color:red; background:blue; font-size:larger;
   } .out { color:black; background:white; font-size:smaller;
   } .click { color:yellow; background:yellow; font-size:12px;
   }
```

```
//HTML代码 <p id="p1" class="out">一大行文字,它们都是普通的文字!</p>
```

```
//JavaScript代码
   window.onload = initAll; function initAll() { var p = document.getElementById("p1");
       p.onclick=clickFn;//这里的事件注册方式除了比行内注册方式少了括号,其它的是一样的
       p.onmouseover = overFn;
       p.onmouseout = outFn;
   } function clickFn() { this.className="click";//这里,this也是可用的
       //注意是className,而不是class
} function overFn() { this.className="over";
   } function outFn() { this.className="out";
   }
```

当然,获取页面元素不止这一种方法.document.getElementsByTagName方法也能获取页面元素,顾名思意,它是通过HTML的标签来获取元素的,而不是ID. 因为一张HTML页面,一个ID名称是唯一的,而标签名则大多数是相同的,所以,getElementsByTagName方法只有一个参数,即一个字符串形式的标签名(tagName),而返回值则是一个类似数组的HTML元素列表

```
    window.onload = initAll;//仍然要写在window.onload事件处理函数中
   function initAll() { var pList = document.getElementsByTagName("P"); //为什么要用大写的P
       alert(pList.length);//与数组相似,length报告有多少个元素,页面上有多少个p标签,就报告多少
       alert(pList[0].innerHTML);//这样来访问第一个p元素
   }
```

另外,对于document.getElementsByTagName方法,还可以传一个"*"号作为参数,以获取页面的所有元素,类似于CSS里面的通配符

```
window.onload = initAll; function initAll() {
   var allThings = document.body.getElementsByTagName("*");
   //可在任何DOM元素上调用getElementsByTagName方法,在body上调用此方法时,body外的标签不会获取到
   alert(allThings.length);//页面上有多少个标签,就报告多少(包含DOCTYPE)
   alert(allThings[3].innerHTML);//这样来访问第四个元素
}
```

其它-javascript:伪协议

伪协议不同于因特网上所真实存在的如http://,https://,ftp://,而是为关联应用程序而使用的.如:tencent://(关联QQ),data:(用base64编码来在浏览器端输出二进制文件),还有就是javascript:

我们可以在浏览地址栏里输入"javascript:alert('JS!');",点转到后会发现,实际上是把javascript:后面的代码当JavaScript来执行,并将结果值返回给当前页面

类似,我们可以在a标签的href属性中使用javascript伪协议

```
<a href="javascript:alert('JS!');"></a> //点击这面的链接,浏览器并不会跳转到任何页面,而是显示-
```

但javascript:伪协议有个问题,它会将执行结果返回给当然的页面

```
<a href="javascript:window.prompt('输入内容将替换当前页面!','');">A</a>
```

解决方法很简单

```
<a href="javascript:window.prompt('输入内容将替换当前页面!','');undefined;">A</a> //将unc
```

尽管javascript伪协议提供了一定的灵活性,但在页面中尽量不要使用!而对于调试JavaScript,javascript伪协议则显得十分有用!

javascript快速入门7--ECMAScript语法基础

ECMAScript的基础概念

熟悉Java、C和Perl这些语言的开发者会发现ECMAScript的语法很容易掌握，因为它借用了这些语言的语法。Java和ECMAScript有一些关键语法特性相同，也有一些完全不同。ECMAScript的基础概念如下：

- 区分大小写。与Java一样，变量、函数名、运算符以及其他一切东西都是区分大小写的，也就是说，变量test不同于变量Test。
- 变量是弱类型的。与Java和C不同，ECMAScript中的变量无特定的类型，定义变量时只用var运算符，可以将它初始化为任意的值。这样可以随时改变变量所存数据的类型（尽管应该避免这样做,但作为Web开发,这确实可以提高效率）。
- 每行结尾的分号可有可无。Java、C和Perl都要求每行代码以分号（;）结束才符合语法。ECMAScript则允许开发者自行决定是否以分号结束一行代码。如果没有分号，ECMAScript就把这行代码的结尾看作该语句的结尾（与Visual Basic和VBScript相似），前提是这样没有破坏代码的语义。最好的代码编写习惯是总加入分号，因为没有分号，有些浏览器就不能正确运行!
- 注释与Java、C和PHP语言的注释相同。ECMAScript借用了这些语言的注释语法。有两种类型的注释——单行注释和多行注释。单行注释以双斜线（//）开头。多行注释以单斜线和星号（/）开头，以星号加单斜线结尾（/）。
- 括号表明代码块。从Java中借鉴的另一个概念是代码块。代码块表示一系列应该按顺序执行的语句，这些语句被封装在左括号（{）和右括号（}）之间。

一些示例如下：

```
var txt = "some string";
   TXT = "other string";//TXT无须声明,可以直接赋值
   alert(txt==TXT);//false
   var str = "string" //分号可有可无
   var hob = "No";var bob = "Yes";//使用分号,可以在一行上写多行语句
   /* 多行注释
   alert("注释中的代码不会被执行") */
   if (txt=="some string") {//代码块
       alert(true);
   }
```

变量

如前所述，ECMAScript中的变量是用var运算符（variable的缩写）加变量名定义的，例如:

```
var test = "Hello!World!";
```

在这个例子中，声明了变量test，并把它的值初始化为"Hello!World!"（字符串）。由于ECMAScript是弱类型的，所以解释程序会为test自动创建一个字符串值，无需明确的类型声明。还可以用一个var语句定义两个或多个变量：

```
var a = "some",b="string"
```

前面的代码定义了变量test，初始值为"some"，还定义了变量test2，初始值为"string"。不过用同一个var语句定义的变量不必具有相同的类型，如下所示：

```
var a=12,b="string";
```

即使a和b属于两种不同的数据类型，在ECMAScript中这样定义也是完全合法的。与Java不同，ECMAScript中的变量并不一定要初始化（它们是在幕后初始化的，将在后面讨论这一点）。因此，下面一行代码也是有效的：

```
var a; //只声明
```

此外，与Java不同的还有变量可以存放不同类型的值。这是弱类型变量的优势。例如，可以把变量初始化为字符串类型的值，之后把它设置为数字值，如下所示：

```
 var test = "string";
    alert(test); //.....若干代码后
    test=123;//更改了类型
    alert(test);
```

这段代码将毫无问题地输出字符串值和数字值。但是，如前所述，使用变量时，好的编码习惯是始终存放相同类型的值。变量名需要遵守两条简单的规则：

- 第一个字符必须是字母、下划线（_）或美圆符号（$）
- 余下的字符可以是下划线、美圆符号或任何字母或数字字符。

下面的变量名都是合法的：

```
 var a; var $a; var $; var _a; var _; var a23;
```

当然，只是因为变量名的语法正确并不意味着就该使用它们。变量还应遵守一条著名的命名规则：

- Camel标记法——首字母是小写的，接下来的单词都以大写字母开头
- Pascal标记法——首字母是大写的，接下来的单词都以大写字母开头
- 匈牙利类型标记法——在以Pascal标记法命名的变量前附加一个小写字母（或小写字母序列），说明该变量的类型。例如，i表示整数，s表示字符串

下面的表列出了用匈牙利类型标记法定义ECMAScript变量使用的前缀：

类型	前缀	示例
数组	a	aValues
布尔型	b	bFound
浮点型（数字）	f	fValue
函数	fn	fnMethod
整型（数字）	i	iValue
对象	o	oType
正则表达式	re	rePattern
字符串	s	sValue
变型（可以是任何类型）	v	vValue

下面是一些命名示例

```
var userName="CJ";//驼峰命名方式
   var UserName="CJ";//Pascal命名方式
   var sUserName="CJ";//匈牙利命名方式
```

ECMAScript另一个有趣的方面（也是与大多数程序设计语言的主要区别）是在使用变量之前不必声明。例如：

```
var str ="some";
   otherStr += str+" "+"string";
   alert(otherStr);
```

在上面的代码中，变量otherStr并没有用var运算符定义，这里只是插入了它，就像已经声明过它。ECMAScript的解释程序遇到未声明过的标识符时，用该变量名创建一个全局变量，并将其初始化为指定的值。这是该语言的便利之处，不过如果不能紧密跟踪变量，这样做也很危险。最好的习惯是像使用其他程序设计语言一样，总是声明所有变量。

关键字

ECMA-262定义了ECMAScript支持的一套关键字（keyword）。这些关键字标识了ECMAScript语句的开头和/或结尾。根据规定，关键字是保留的，不能用作变量名或函数名。下面是ECMAScript关键字的完整列表：

break	else	new	var
case	finally	return	void
catch	for	switch	while
continue	function	this	with
default	if	throw	
delete	in	try	
do	instanceof	typeof	

如果把关键字用作变量名或函数名，可能得到诸如“Identifier expected”（应该有标识符,缺少标识符）这样的错误消息。

保留字

保留字是对于JavaScript有特殊含义的单词。因此，不能将它们用作变量名或函数名。也就是说，它们可能是JavaScript未来版本中的命令。现在就应该避免使用它们，以免在新版本发布时不得不修改代码。如果将保留字用作变量名或函数名，那么除非将来的浏览器实现了该保留字，否则很可能收不到任何错误消息。当浏览器将其实现后，该单词将被看作关键字，如此将出现关键字错误。

ECMAScript 3为以后保留的单词：

abstract	final	protected
boolean	float	public
byte	goto	short
char	implements	static
class	import	super
const	int	synchronized
debugger	interface	throws
double	long	transient
enum	native	volatile
export	package	extends
private		

ECMAScript 4：ECMAScript4现在还没有什么实现.ECMAScript4中,下面的不再是保留字了,但也应该尽量不要使用它们

```
boolean final short byte float static char int double long
```

下面的被加入了保留字

```
as namespace use false true null is
```

javascript快速入门8--值,类型与类型转换

原始值和引用值

在ECMAScript中，变量可以存放两种类型的值，即原始值和引用值。

- 原始值（primitive value）是存储在栈（stack）中的简单数据段，也就是说，它们的值直接存储在变量访问的位置。
- 引用值（reference value）是存储在堆（heap）中的对象，也就是说，存储在变量处的值是一个指针（point），指向存储对象的内存处。

为变量赋值时，ECMAScript的解释程序必须判断该值是原始类型的，还是引用类型的。要实现这一点，解释程序则需尝试判断该值是否为ECMAScript的原始类型之一，即Undefined、Null、Boolean和String型。由于这些原始类型占据的空间是固定的，所以可将它们存储在较小的内存区域——栈中。这样存储便于迅速查寻变量的值。

在许多语言中，字符串都被看作引用类型，而非原始类型，因为字符串的长度是可变的。**ECMAScript**打破了这一传统。

如果一个值是引用类型的，那么它的存储空间将从堆中分配。由于引用值的大小会改变，所以不能把它放在栈中，否则会降低变量查寻的速度。相反，放在变量的栈空间中的值是该对象存储在堆中的地址。地址的大小是固定的，所以把它存储在栈中对变量性能无任何负面影响,如图:



堆栈就像我们的书一样,堆对应着书的正文内容,而栈对应的是书的目录,章节标题是简短的,所以在目录里面可以放,而文章的内容则不能放在目录里,我们只需要在目录放一个对应文章的标题和页码.这样,即可以在一本书中放大量的内容,又因为存在目录,可以快速查找内容!

原始类型

如前所述，ECMAScript有5种原始类型（primitive type），即Undefined、Null、Boolean、Number和String。ECMA-262把术语类型（type）定义为值的一个集合，每种原始类型定义了它包含的值的范围及其字面量表示形式。ECMAScript提供了typeof运算符来判断一个值是否在某种类型的范围内。可以用这种运算符判断一个值是否表示一种原始类型；如果它是原始类型，还可以判断它表示哪种原始类型。

typeof运算符有一个参数，即要检查的变量或值。例如：

```
var str = "some thing";
   alert(typeof str);//输出string
   var num =123;
   alert(typeof(num));//输出number,typeof运算符可以像函数一样使用
   //这主要运用于一些复杂的表达的以解决优先级问题
   alert(typeof num*3);//输出NaN,因为typeof优先于*
   alert(typeof(num*3));//输出number
```

对变量或值调用typeof运算符将返回下列值之一：

- "undefined"，如果变量是Undefined型的
- "boolean"，如果变量是Boolean型的
- "number"，如果变量是Number型的
- "string"，如果变量是String型的
- "object"，如果变量是一种引用类型或Null类型的

你也许会问，为什么**typeof**运算符对于**null**值会返回**"object"**。这实际上是**JavaScript**最初实现中的一个错误，然后被**ECMAScript**沿用了。现在，**null**被认为是对象的占位符，从而解释了这一矛盾，但从技术上来说，它仍然是原始值。

如前所述，Undefined类型只有一个值，即undefined。当声明的变量未初始化时，该变量的默认值是undefined

```
var a;
   alert(a);//输出undefined
   alert(typeof a);//输出undefined
   alert(a==undefined);//true
```

注意，值undefined并不同于未定义的值。但是，typeof运算符并不真正区分这两种值。考虑下面的代码：

```
var tmp;
   alert(tmp);//undefined
   alert(typeof tmp);//undefined
   alert(typeof tmp2);//undefined
```

前面的代码对两个变量输出的都是"undefined"，即使只有变量tmp2是未定义的。如果不用typeof运算符，就对tmp2使用其他运算符，这将引起错误，因为那些运算符只能用于已定义的变量。例如，下面的代码将引发错误：

```
   alert(tmp3==undefined);//出错,因为变量tmp3未声明
```

当函数无明确返回值时，返回的也是值undefined，如下所示：

```
function tmp() {}//空函数,没有返回值
   alert(tmp());//undefined
```

另一种只有一个值的类型是Null，它只有一个专用值null，即它的字面量。值undefined实际上是从值null派生来的，因此ECMAScript把它们定义为相等的。但它们并不是严格相等的,因为它们是不同类型!

```
   alert(null==undefined);//输出true
```

尽管这两个值相等，但它们的含义不同。undefined是声明了变量但未对其初始化时赋予该变量的值，null则用于表示尚未存在的对象。如果函数或方法要返回的是对象，那么找不到该对象时，返回的通常是null。

```
var obj = document.getElementById("tmp");//页面上没有ID为tmp的元素
   alert(obj);//null
```

Boolean类型是ECMAScript中最常用的类型之一。它有两个值true和false（即两个Boolean字面量）。即使false不等于0，0也可以在必要时被转换成false，这样在Boolean语句中使用两者都是安全的。

```
var bool=true;
   bool=false;
```

ECMA-262中定义的最特殊的类型是Number型。这种类型既可以表示32位的整数，还可以表示64位的浮点数。直接输入的（而不是从另一个变量访问的）任何数字都被看作Number型的字面量。例如，下面的代码声明了存放整数值的变量，它的值由字面量55定义：

```
var num =55;
```

整数也可以被表示为八进制（以8为底）或十六进制（以16为底）的字面量。八进制字面量的首数字必须是0，其后的数字可以是任何八进制数字（0到7），如下面代码所示：

```
var num=020;//八进制10为十进制的16
   alert(num);//虽然我们用八进制方法表示一个数,但输出时,系统会自动输出它的十进制表现形式
```

要创建十六进制的字面量，首位数字必须为0，其后接字母x，然后是任意的十六进制数字（0到9和A到F）。这些字母可以是大写的，也可以是小写的。例如：

```
var num=0x12;//十进制18
   num=0xab;//171
   alert(num);//输出的仍是十进制的171
```

注意!尽管所有整数都可表示为八进制或十六进制的字面量，但所有数学运算返回的都是十进制结果!

要定义浮点值，必须包括小数点和小数点后的一位数字（例如，用1.0而不是1）。这被看作浮点数字面量。例如：

```
var num = 1;//整数
   var num2=1.0;//浮点数
```

对于非常大或非常小的数，可以用科学记数法表示浮点值。采用科学记数法，可以把一个数表示为数字（包括十进制数字）加e（或E），后面加乘以10的倍数。不明白？下面是一个示例：

```
var num = 3E10;
   alert(num);//30000000000
   num=3.45E2;
   alert(num);//345
```

也可用科学记数法表示非常小的数，例如0.00000000000000003可以表示为3-e17（这里，10被升到-17次幂，意味着需要被10除17次）。ECMAScript默认把具有6个或6个以上前导0的浮点数转换成科学记数法。

几个特殊值也被定义为Number类型的。前两个是Number.MAX*VALUE*和*Number.MIN*
VALUE，它们定义了Number值集合的外边界。所有ECMAScript数都必须在这两个值之间。不过计算生成的数值结果可以不落在这两个数之间。当计算生成的数大于Number.MAX_VALUE时，它将被赋予值Number.POSITIVE_INFINITY，意味着不再有数字值。同样，生成的数值小于Number.MIN_VALUE的计算也会被赋予值Number.NEGATIVE_INFINITY，也意味着不再有数字值。如果计算返回的是无穷大值，那么生成的结果不能再用于其他计算。

事实上，有专门的值表示无穷大，即Infinity。Number.POSITIVE_INFINITY的值为Infinity，Number.NEGATIVE_INFINITY的值为-Infinity。

```
    alert(Number.MAX_VALUE*2 == Infinity);//true
    alert(Number.NEGATIVE_INFINITY == -Infinity);//true
```

由于无穷大数可以是正数也可以是负数，所以可用一个方法判断一个数是否是有穷的（而不是单独测试每个无穷数）。可以对任何数调用isFinit()方法，以确保该数不是无穷大。例如：

```
var a = 9E99999999999999999999999999999999;//已经超过范围了
    if (isFinite(a)) {
        alert("一个有穷数!");
    } else {
        alert("一个无穷数!");
    }
```

最后一个特殊值是NaN，表示非数（Not a Number）。NaN是个奇怪的特殊值。一般说来，这种情况发生在类型（String、Boolean等）转换失败时。例如，要把单词blue转换成数值就会失败，因为没有与之等价的数值。与无穷大值一样，NaN也不能用于算术计算。NaN的另一个奇特之处在于，它与自身不相等，这意味着下面的代码将返回false：

```
alert(NaN);
        alert(!!NaN);
        alert(NaN == NaN);
```

出于这种原因，不推荐使用NaN值本身。函数isNaN()会做得相当好：

```
    alert(isNaN(NaN));//true
    alert(isNaN(123));//false
```

String类型的独特之处在于，它是唯一没有固定大小的原始类型。可以用字符串存储0或更多的Unicode字符，由16位整数表示（Unicode是一种国际字符集，本书后面将讨论它）。字符串中每个字符都有特定的位置，首字符从位置0开始，第二个字符在位置1，依此类推。这意味着字符串中的最后一个字符的位置一定是字符串的长度减1

字符串字面量是由双引号（"）或单引号（'）声明的。与Java不同的是，双引号用于声明字符串，单引号用于声明字符。但是，由于ECMAScript没有字符类型，所以可使用这两种表示法中的任何一种。

String类型还包括几种字符字面量，Java、C和Perl的开发者应该对此非常熟悉。下表列出了ECMAScript的字符字面量：

字面量	含义
\n	换行
\t	制表符
\b	空格
\r	回车
\f	换页符
\	反斜杠
\'	单引号
\"	双引号
\0nnn	八进制代码nnn（n是0到7中的一个八进制数字）表示的字符
\xnn	十六进制代码nn（n是0到F中的一个十六进制数字）表示的字符
\unnnn	十六进制代码nnnn（n是0到F中的一个十六进制数字）表示的Unicode字符

类型转换

所有程序设计语言最重要的特征之一是具有进行类型转换的能力，ECMAScript给开发者提供了大量简单的转换方法。大多数类型具有进行简单转换的方法，还有几个全局方法可以用于更复杂的转换。无论哪种情况，在ECMAScript中，类型转换都是简短的一步操作。

ECMAScript的Boolean值、数字和字符串的原始值的有趣之处在于它们是伪对象，这意味着它们实际上具有属性和方法。例如，要获得字符串的长度，可以采用下面的代码：

```
var str ="some";
   alert(str.length);
```

尽管"blue"是原始类型的字符串，它仍然具有属性length，用于存放该字符串的大小。总而言之，3种主要的原始值Boolean值、数字和字符串都有toString()方法，可以把它们的值转换成字符串。

也许你会问，“字符串还有**toString()**方法，这不是多余的吗？”是的，的确如此，不过**ECMAScript**定义所有对象都有**toString()**方法，无论它是伪对象，还是真的对象。因为**String**类型属于伪对象，所以它一定有**toString()**方法。

Boolean型的toString()方法只是输出"true"或"false"，结果由变量的值决定：

```
var bool=true;
   alert(bool.toString());
```

Number类型的toString()方法比较特殊，它有两种模式，即默认模式和基模式。采用默认模式，toString()方法只是用相应的字符串输出数字值（无论是整数、浮点数还是科学记数法），如下所示：

```
var num=10;
   alert(num.toString());
   num=10.0;
   alert(num.toString());//仍然是10
```

在默认模式中，无论最初采用什么表示法声明数字，**Number**类型的**toString()**方法返回的都是数字的十进制表示。因此，以八进制或十六进制字面量形式声明的数字输出时都是十进制形式的。

采用Number类型的toString()方法的基模式，可以用不同的基输出数字，例如二进制的基是2，八进制的基是8，十六进制的基是16。基只是要转换成的基数的另一种叫法而已，它是toString()方法的参数：

```
var num=10;
   alert(num.toString());//10
   alert(num.toString(2));//1010
   alert(num.toString(8));//12
   alert(num.toString(16));//A
   //对数字调用toString(10)与调用toString()相同，它们返回的都是该数字的十进制形式。
   alert(num.toString(10));
```

ECMAScript提供了两种把非数字的原始值转换成数字的方法，即parseInt()和parseFloat()。正如你可能想到的，前者把值转换成整数，后者把值转换成浮点数。只有对String类型调用这些方法，它们才能正确运行；对其他类型返回的都是NaN。

```
   alert(parseInt("12.23"));//12
   alert(parseFloat("12.45"));//12.45
   alert(parseFloat("12.23.34"));//12.23
   alert(parseFloat("abc"));//NaN
   alert(parseInt(true));//NaN
```

在判断字符串是否是数字值前，parseInt()和parseFloat()都会仔细分析该字符串。parseInt()方法首先查看位置0处的字符，判断它是否是个有效数字；如果不是，该方法将返回NaN，不再继续执行其他操作。但如果该字符是有效数字，该方法将查看位置1处的字符，进行同样的测试。这一过程将持续到发现非有效数字的字符为止，此时parseInt()将把该字符之前的字符串转换成数字。例如，如果要把字符串"1234blue"转换成整数，那么parseInt()将返回1234，因为当它检测到字符b时，就会停止检测过程。字符串中包含的数字字面量会被正确转换为数字，因此字符串"0xA"会被正确转换为数字10。不过，字符串"22.5"将被转换成22，因为对于整数来说，小数点是无效字符。

parseInt()方法还有基模式，可以把二进制、八进制、十六进制或其他任何进制的字符串转换成整数。基是由parseInt()方法的第二个参数指定的，所以要解析十六进制的值，需如下调用parseInt()方法：

```
    alert(parseInt("AB",16));//171
    alert(parseInt("10",2));//2
    alert(parseInt("10",8));//8
```

如果十进制数包含前导0，那么最好采用基数10，这样才不会意外地得到八进制的值。例如：

```
var str ="010";
    alert(parseInt(str));//8
    alert(parseInt(str,10));//10
```

parseFloat()没有基模式

强制类型转换

还可使用强制类型转换（type casting）处理转换值的类型。使用强制类型转换可以访问特定的值，即使它是另一种类型的。ECMAScript中可用的3种强制类型转换如下：

- Boolean(value)——把给定的值转换成Boolean型
- Number(value)——把给定的值转换成数字（可以是整数或浮点数）
- String(value)——把给定的值转换成字符串

用这三个函数之一转换值，将创建一个新值，存放由原始值直接转换成的值。这会造成意想不到的后果。

当要转换的值是至少有一个字符的字符串、非0数字或对象时，Boolean()函数将返回true。如果该值是空字符串、数字0、undefined或null，它将返回false。可以用下面的代码段测试Boolean型的强制类型转换。

```
var b1 = Boolean("");        //false - 空字符串
    var b2 = Boolean("hello");        //true - 非空字符串
    var b1 = Boolean(50);        //true - 非零数字
    var b1 = Boolean(null);        //false - null
    var b1 = Boolean(0);        //false - 零
    var b1 = Boolean(new object());    //true - 对象
```

Number() 函数的强制类型转换与 parseInt() 和 parseFloat() 方法的处理方式相似，只是它转换的是整个值，而不是部分值。

parseInt() 和 parseFloat() 方法只转换第一个无效字符之前的字符串，因此 "1.2.3" 将分别被转换为 "1" 和 "1.2"。用 Number() 进行强制类型转换，"1.2.3" 将返回 NaN，因为整个字符串值不能转换成数字。如果字符串值能被完整地转换，Number() 将判断是调用 parseInt() 方法还是 parseFloat() 方法。

下表说明了对不同的值调用 Number() 方法会发生的情况：

用法	结果
Number(false)	0
Number(true)	1
Number(undefined)	NaN
Number(null)	0
Number("1.2")	1.2
Number("12")	12
Number("1.2.3")	NaN
Number(new Object())	NaN
Number(50)	50

最后一种强制类型转换方法 String() 是最简单的，因为它可把任何值转换成字符串。要执行这种强制类型转换，只需要调用作为参数传递进来的值的 toString() 方法，即把 12 转换成 "12"，把 true 转换成 "true"，把 false 转换成 "false"，以此类推。强制转换成字符串和调用 toString() 方法的唯一不同之处在于，对 null 和 undefined 值强制类型转换可以生成字符串而不引发错误：

```
var s1 = String(null);    //"null"
   var oNull = null; var s2 = oNull.toString();    //会引发错误
```

在处理 ECMAScript 这样的弱类型语言时，强制类型转换非常有用，不过应该确保使用值的正确。

javascript快速入门9--引用类型

引用类型通常叫做类（class），也就是说，遇到引用值，所处理的就是对象。

注意：从传统意义上来说，**ECMAScript** 并不真正具有类。事实上，除了说明不存在类，在 **ECMA-262** 中根本没有出现“类”这个词。**ECMAScript** 定义了“对象定义”，逻辑上等价于其他程序设计语言中的类

Object

对象是由 new 运算符加上要实例化的对象的名字创建的。例如，下面的代码创建 Object 对象的实例：

```
var obj = new Object();
```

这种语法与 Java 语言的相似，不过当有不止一个参数时，ECMAScript 要求使用括号。如果没有参数，如以下代码所示，括号可以省略：

```
var o = new Object;
```

注意：尽管括号不是必需的，但是为了避免混乱，最好使用括号。

Object 对象自身用处不大，不过在了解其他类之前，还是应该了解它。因为 ECMAScript 中的 Object 对象与 Java 中的 java.lang.object 相似，ECMAScript 中的所有对象都由这个对象继承而来，Object 对象中的所有属性和方法都会出现在其他对象中，所以理解了 Object 对象，就可以更好地理解其他对象。

Object 对象具有下列属性：

constructor

对创建对象的函数的引用（指针）。对于 Object 对象，该指针指向原始的 Object() 函数。

prototype

对该对象的对象原型的引用。对于所有的对象，它默认返回 Object 对象的一个实例。

Object 对象还具有几个方法：

hasOwnProperty(property)

判断对象是否有某个特定的属性。必须用字符串指定该属性。（例如，o.hasOwnProperty("name")）

isPrototypeOf(object)

判断该对象是否为另一个对象的原型。

propertyIsEnumerable(property)

判断给定的属性是否可以用 for...in 语句进行枚举。

toString()

返回对象的原始字符串表示。对于 Object 对象，ECMA-262 没有定义这个值，所以不同的 ECMAScript 实现具有不同的值。

valueOf()

返回最适合该对象的原始值。对于许多对象，该方法返回的值都与 ToString() 的返回值相同。

注意：上面列出的每种属性和方法都会被其他对象覆盖。

Boolean

Boolean 对象是 Boolean 原始类型的引用类型。要创建 Boolean 对象，只需要传递 Boolean 值作为参数：

```
var oBooleanObject = new Boolean(true);
```

Boolean 对象将覆盖 Object 对象的 ValueOf() 方法，返回原始值，即 true 和 false。ToString() 方法也会被覆盖，返回字符串 "true" 或 "false"。遗憾的是，在 ECMAScript 中很少使用 Boolean 对象，即使使用，也不易理解。问题通常出现在 Boolean 表达式中使用 Boolean 对象时。例如：

```
var oFalseObject = new Boolean(false); var bResult = oFalseObject && true;    //输出 true
```

在这段代码中，用 false 值创建 Boolean 对象。然后用这个值与原始值 true 进行 AND 操作。在 Boolean 运算中，false 和 true 进行 AND 操作的结果是 false。不过，在这行代码中，计算的是 oFalseObject，而不是它的值 false。正如前面讨论过的，在 Boolean 表达式中，所有对象都会被自动转换为 true，所以 oFalseObject 的值是 true。然后 true 再与 true 进行 AND 操作，结果为 true。

注意：虽然你应该了解 **Boolean** 对象的可用性，不过最好还是使用 **Boolean** 原始值，避免发生这一节提到的问题。

Number

Number 对象是 Number 原始类型的引用类型。要创建 Number 对象，采用下列代码：

```
var oNumberObject = new Number(68);
```

Number.MAX_VALUE,Number.MIN_VALUE等特殊值都是 **Number** 对象的静态属性。

要得到数字对象的 Number 原始值，只需要使用 valueOf() 方法：

```
var iNumber = oNumberObject.valueOf();
```

当然，Number 类也有 toString() 方法.除了从 Object 对象继承的标准方法外，Number 对象还有几个处理数值的专用方法:

toFixed() 方法返回的是具有指定位数小数的数字的字符串表示。例如：

```
var oNumberObject = new Number(68);
   alert(oNumberObject.toFixed(2));  //输出 "68.00"
```

在这里，toFixed() 方法的参数是 2，说明应该显示两位小数。该方法返回 "68.00"，空的字符串位由 0 来补充。对于处理货币的应用程序，该方法非常有用。toFixed() 方法能表示具有 0 到 20 位小数的数字，超过这个范围的值会引发错误。

与格式化数字相关的另一个方法是 toExponential()，它返回的是用科学计数法表示的数字的字符串形式。与 toFixed() 方法相似，toExponential() 方法也有一个参数，指定要输出的小数的位数。例如：

```
var oNumberObject = new Number(68);
   alert(oNumberObject.toExponential(1));  //输出 "6.8e+1"
```

这段代码的结果是 "6.8e+1"，前面解释过，它表示 6.8x101。问题是，如果不知道要用哪种形式（预定形式或指数形式）表示数字怎么办？可以用 toPrecision() 方法。

toPrecision() 方法根据最有意义的形式来返回数字的预定形式或指数形式。它有一个参数，即用于表示数的数字总数（不包括指数）。例如:

```
var oNumberObject = new Number(68);
   alert(oNumberObject.toPrecision(2));  //输出 "68"
```

当然，输出的是 "68"，因为这正是该数的准确表示。不过，如果指定的位数多于需要的位数又如何呢？

```
var oNumberObject = new Number(68);
   alert(oNumberObject.toPrecision(3));  //输出 "68.0"
```

在这种情况下，toPrecision(3) 等价于 toFixed(1)，输出的是 "68.0"。toFixed()、toExponential() 和 toPrecision() 方法都会进行舍入操作，以便用正确的小数位数正确地表示一个数。

String

String 对象是 String 原始类型的对象表示法，它是以下方式创建的：

```
var oStringObject = new String("hello world");
```

String 对象的 valueOf() 方法和 toString() 方法都会返回 String 类型的原始值：

```
   alert(oStringObject.valueOf() == oStringObject.toString());    //输出 "true"
```

String 对象具有属性 length，它是字符串中的字符个数：

```
var oStringObject = new String("hello world");
   alert(oStringObject.length); //输出 "11"
```

注意，即使字符串包含双字节的字符（与 **ASCII** 字符相对，**ASCII** 字符只占用一个字节），每个字符也只算一个字符。

String 对象还拥有大量的方法。

localeCompare() 方法,对字符串进行排序。该方法有一个参数 - 要进行比较的字符串，返回的是下列三个值之一：

- 如果 String 对象按照字母顺序排在参数中的字符串之前，返回负数。
- 如果 String 对象等于参数中的字符串，返回 0
- 如果 String 对象按照字母顺序排在参数中的字符串之后，返回正数。

如果返回负数，那么最常见的是 **-1**，不过真正返回的是由实现决定的。如果返回正数，那么同样的，最常见的是 **1**，不过真正返回的是由实现决定的。而且对于中文字符,虽然表面上看是按拼音排序的,但并不一定!事实上,它们按**Unicode**编码排序,小的排在前面,大的排在后面

示例如下：

```
var oStringObject = new String("yellow");
   alert(oStringObject.localeCompare("brick"));        //输出 "1"
   alert(oStringObject.localeCompare("yellow"));        //输出 "0"
   alert(oStringObject.localeCompare("zoo"));        //输出 "-1"
```

再强调一次，由于返回的值是由实现决定的，所以最好以下面的方式调用 localeCompare() 方法：

```
var oStringObject1 = new String("yellow"); var oStringObject2 = new String("brick"); var
      alert(oStringObject2 + " 排在字符串 " + oStringObject1 +"后面");
   } else if (iResult > 0) {
      alert(oStringObject2 + " 排在字符串 " + oStringObject1+"前面");
   } else {
      alert("两个字符串排序是一样的！");
   }
```

localeCompare() 方法的独特之处在于，实现所处的区域（locale，兼指国家/地区和语言）确切说明了这种方法运行的方式。在美国，英语是 ECMAScript 实现的标准语言，localeCompare() 是区分大小写的，大写字母在字母顺序上排在小写字母之后。不过，在其他区域，情况可能并非如此。

ECMAScript 提供了两种方法从子串创建字符串值，即 slice() 和 substring()。这两种方法返回的都是要处理的字符串的子串，都接受一个或两个参数。第一个参数是要获取的子串的起始位置，第二个参数（如果使用的话）是要获取子串终止前的位置（也就是说，获取终止位置处的字符不包括在返回的值内）。如果省略第二个参数，终止位就默认为字符串的长度。slice() 和 substring() 方法都不改变 String 对象自身的值。它们只返回原始的 String 值，保持 String 对象不变。

```
var oStringObject = new String("hello world");
   alert(oStringObject.slice(3));        //输出 "lo world"
   alert(oStringObject.substring(3));        //输出 "lo world"
   alert(oStringObject.slice(3, 7));        //输出 "lo w"
   alert(oStringObject.substring(3, 7));    //输出 "lo w"
```

为什么有两个功能完全相同的方法呢？事实上，这两个方法并不完全相同，不过只在参数为负数时，它们处理参数的方式才稍有不同。对于负数参数，slice() 方法会用字符串的长度加上参数，substring() 方法则将其作为 0 处理（也就是说将忽略它）。例如：

```
var oStringObject = new String("hello world");
   alert(oStringObject.slice(-3));        //输出 "rld"
   alert(oStringObject.substring(-3));    //输出 "hello world"
   alert(oStringObject.slice(3, -4));        //输出 "lo w"
   alert(oStringObject.substring(3, -4));    //输出 "hel"
```

最后一套要讨论的方法涉及大小写转换。有 4 种方法用于执行大小写转换，即

- toLowerCase()
- toLocaleLowerCase()
- toUpperCase()
- toLocaleUpperCase()

从名字上可以看出它们的用途，前两种方法用于把字符串转换成全小写的，后两种方法用于把字符串转换成全大写的。toLowerCase() 和 toUpperCase() 方法是原始的，是以 java.lang.String 中相同方法为原型实现的。toLocaleLowerCase() 和 toLocaleUpperCase()

方法是基于特定的区域实现的（与 localeCompare() 方法相同）。在许多区域中，区域特定的方法都与通用的方法完全相同。不过，有几种语言对 Unicode 大小写转换应用了特定的规则（例如土耳其语），因此必须使用区域特定的方法才能进行正确的转换。

String 对象的所有属性和方法都可应用于 **String** 原始值上，因为它们是伪对象。

String对象方法参考

方法	描述
charAt()	返回在指定位置的字符。
charCodeAt()	返回在指定的位置的字符的 Unicode 编码。
concat()	连接字符串。
fromCharCode()	从字符编码创建一个字符串。
indexOf()	检索字符串。
lastIndexOf()	从后向前搜索字符串。
localeCompare()	用本地特定的顺序来比较两个字符串。
match()	找到一个或多个正则表达式的匹配。
replace()	替换与正则表达式匹配的子串。
search()	检索与正则表达式相匹配的值。
slice()	提取字符串的片断，并在新的字符串中返回被提取的部分。
split()	把字符串分割为字符串数组。
substr()	从起始索引号提取字符串中指定数目的字符。
substring()	提取字符串中两个指定的索引号之间的字符。
toLocaleLowerCase()	把字符串转换为小写。
toLocaleUpperCase()	把字符串转换为大写。
toLowerCase()	把字符串转换为小写。
toUpperCase()	把字符串转换为大写。
toString()	返回字符串。
valueOf()	返回某个字符串对象的原始值。

Global

Global是一个最为特殊的对象,因为它实际上根本不存在!如果尝试输出Global对象将出错:

```
alert(Global);
```

但Global确实是一个对象,要理解这一点要先理解ECMAScript中一个概念,即在ECMAScript中不存在独立的函数,所有的函数必须是某个对象的方法. 像之前所使用的isNaN(),parseInt()看上去像独立的函数,其实它们都是Global对象的方法,Global对象的方法还不止这些,如用于编解码URI的函数:encodeURI与decodeURI方法

```
var url = "http://www.g.cn/trend page";
   alert(encodeURI(url));//该方法会将一些不能用在URI中的字符进行编码
   //上面的空格将会被替换成%20,与之对应的decodeURI则是用来解码的
   alert(decodeURI(encodeURI(url)));//原样输出
```

与encodeURI和decodeURI相似的encodeURIComponent和decodeURIComponent . 这两个方法不但对空格之类的字符进行编码,还对"/"和":"这些URL中的特殊字符进行编码

```
var url = "http://www.g.cn";
   alert(encodeURIComponent(url)); //输出http://%3A%2F%2Fwww.g.cn
```

一般情况下要把一些字符串作为QueryString值添加到URL后面时,使用encodeURIComponent更安全些. encodeURI之类的方法代替了以前BOM提供的escape与unescape之类的方法,URI方法更可取,因为escape方法只能对ASCII字符进行编码,而URI方法则可以对所有的Unicode字符进行编码. 应该总是使用URI方法,避免使用escape方法!

Global对象不但有方法,还有一些属性:如undefined,NaN,Infinity这些特殊值都是Global对象的属性,还有所有的内置引用类型的构造函数(Array,Date等)都是的,只是不能通过Global.Array这样方法输出来,它们都被映射到了window对象上了.

javascript快速入门10--运算符,语句

一元运算符

一元运算符只有一个参数，即要操作的对象或值。它们是 ECMAScript 中最简单的运算符。

delete 运算符删除对以前定义的对象属性或方法的引用。例如：

```
var obj = new Object();
   obj.name = "David";
   alert(obj.name); //输出 "David"
   delete obj.name;
   alert(obj.name); //输出 "undefined"
```

delete 运算符不能删除开发者未定义的属性和方法。例如，下面的代码是没什么效果的：

```
delete obj.toString;
```

即使 toString 是有效的方法名，这行代码也会引发错误，因为 toString() 方法是原始的 ECMAScript 方法，不是开发者定义的。

delete 还可以用来删除数组中的元素(这没什么奇怪的,数组和对象是相通的)

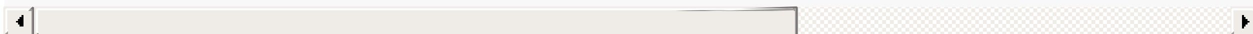
```
var arr = [2,4,6,8,10]; delete arr[2];
   alert(arr[2]);//undefined
```

void 运算符对任何值返回 undefined。该运算符通常用于避免输出不应该输出的值，例如，从 HTML 的 <a> 元素调用 JavaScript 函数时。要正确做到这一点，函数不能返回有效值，否则浏览器将清空页面，只显示函数的结果。例如：

```
<a href="javascript:window.open('about:blank')">Click me</a>
```

如果把这行代码放入 HTML 页面，点击其中的链接，即可看到屏幕上显示 "[object]"。TIY这是因为 window.open() 方法返回了新打开的窗口的引用。然后该对象将被转换成要显示的字符串。要避免这种效果，可以用 void 运算符调用 window.open() 函数：

```
<a href="javascript:void(window.open('about:blank'))">A</a> //这使 window.open() 调用返回
```



没有返回值的函数真正返回的都是 **undefined**。

直接从 C（和 Java）借用的两个运算符是前增量运算符和前减量运算符。所谓前增量运算符，就是数值上加 1，形式是在变量前放两个加号（++）：

```
var iNum = 10; ++iNum; //实际上等价于
   var iNum = 10;
   iNum = iNum + 1;
```

同样，前减量运算符是从数值上减 1，形式是在变量前放两个减号（--）：

```
var iNum = 10; --iNum;
```

在使用前缀式运算符时，注意增量和减量运算符都发生在计算表达式之前。考虑下面的例子：

```
var iNum = 10; --iNum;
   alert(iNum); //输出 "9"
   alert(--iNum);    //输出 "8"
   alert(iNum);    //输出 "8"
```

第二行代码对 iNum 进行减量运算，第三行代码显示的结果是（"9"）。第四行代码又对 iNum 进行减量运算，不过这次前减量运算和输出操作出现在同一个语句中，显示的结果是 "8"。为了证明已实现了所有的减量操作，第五行代码又输出一次"8"。在算术表达式中，前增量和前减量运算符的优先级是相同的，因此要按照从左到右的顺序计算之。例如：

```
var iNum1 = 2; var iNum2 = 20; var iNum3 = --iNum1 + ++iNum2;    //等于 "22"
   var iNum4 = iNum1 + iNum2;//等于 "22"
```

还有两个直接从 C（和 Java）借用的运算符，即后增量运算符和后减量运算符。后增量运算符也是给数值上加 1，形式是在变量后放两个加号（++）：

```
var iNum = 10;
   iNum++;
```

后减量运算符是从数值上减 1，形式为在变量后加两个减号（--）：

```
var iNum = 10;
   iNum--;
```

与前缀式运算符不同的是，后缀式运算符是在计算过包含它们的表达式后才进行增量或减量运算的。考虑以下的例子：

```
var iNum = 10;
   iNum--;
   alert(iNum); //输出 "9"
   alert(iNum--);    //输出 "9"
   alert(iNum);    //输出 "8"
```

在算术表达式中，后增量和后减量运算符的优先级是相同的，因此要按照从左到右的顺序计算之。例如：

```
var iNum1 = 2; var iNum2 = 20; var iNum3 = iNum1-- + iNum2++;    //等于 "22"
   var iNum4 = iNum1 + iNum2;        //等于 "22"
```

在前面的代码中，iNum3 等于 22，因为表达式要计算的是 2 + 20。变量 iNum4 也等于 22，不过它计算的是 1 + 21，因为增量和减量运算都在给 iNum3 赋值后才发生。

大多数人都熟悉一元加法和一元减法，它们在 ECMAScript 中的用法与您高中数学中学到的用法相同。一元加法本质上对数字无任何影响,只用来表示正数,但不能将负数转换成正数

```
var num=-10;
   alert(+num);//-10
```

但并不是一元加法就一无用处.由于这类的运算符只能作用于数字,因些,当将这些运算符运用于字符串时,可以起到将字符串转换成数字的妙效!

```
var str ="123";
   str = +str;
   alert(typeof str);//number
```

一元减法也有这样的效用,但同时,它还会对数字求负!

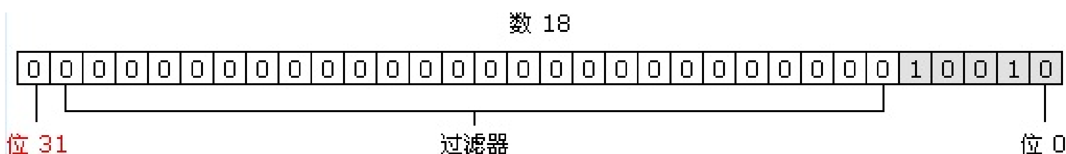
位运算符

位运算符是在数字底层（即表示数字的 32 个数位）进行操作的。

ECMAScript 整数有两种类型，即有符号整数（允许用正数和负数）和无符号整数（只允许用正数）。在 ECMAScript 中，所有整数字面量默认都是有符号整数!

有符号整数使用 31 位表示整数的数值，用第 32 位表示整数的符号，0 表示正数，1 表示负数。数值范围从 -2147483648 到 2147483647。

可以以两种不同的方式存储二进制形式的有符号整数，一种用于存储正数，一种用于存储负数。正数是以真二进制形式存储的，前 31 位中的每一位都表示 2 的幂，从第 1 位（位 0）开始，表示 20，第 2 位（位 1）表示 21。没用到的位用 0 填充，即忽略不计。例如，下面展示的是数 18 的表示法。



18 的二进制版本只用了前 5 位，它们是这个数字的有效位。把数字转换成二进制字符串，就能看到有效位：

```
var iNum = 18;
   alert(iNum.toString(2));    //输出 "10010"
```

这段代码只输出 "10010"，而不是 18 的 32 位表示。其他的数位并不重要，因为仅使用前 5 位即可确定这个十进制数值。如下图所示：

1	0	0	1	0

$(2^4x1) + (2^3x0) + (2^2x0) + (2^1x1) + (2^0x0)$

16 + 0 + 0 + 2 + 0

18

负数也存储为二进制代码，不过采用的形式是二进制补码。计算数字二进制补码的步骤有三步：

1. 确定该数字的非负版本的二进制表示（例如，要计算 -18的二进制补码，首先要确定 18 的二进制表示）
2. 求得二进制反码，即要把 0 替换为 1，把 1 替换为 0
3. 在二进制反码上加 1

要确定 -18 的二进制表示，首先必须得到 18 的二进制表示，如下所示：

```
0000 0000 0000 0000 0000 0000 0001 0010
```

接下来，计算二进制反码，如下所示：

```
1111 1111 1111 1111 1111 1111 1110 1101
```

最后，在二进制反码上加 1，如下所示：

```
1111 1111 1111 1111 1111 1111 1110 1101
                                                                   1
-----------------------------------------------------------
1111 1111 1111 1111 1111 1111 1110 1110
```

因此，-18 的二进制表示即 1111 1111 1111 1111 1111 1111 1110 1110。记住，在处理有符号整数时，开发者不能访问 31 位。

然而，把负整数转换成二进制字符串后，ECMAScript 并不以二进制补码的形式显示，而是用数字绝对值的标准二进制代码前面加负号的形式输出。例如：

```
var iNum = -18;
   alert(iNum.toString(2));    //输出 "-10010"
```

这段代码输出的是 "-10010"，而非二进制补码，这是为避免访问位 31。为了简便，ECMAScript 用一种简单的方式处理整数，使得开发者不必关心它们的用法。另一方面，无符号整数把最后一位作为另一个数位处理。在这种模式中，第 32 位不表示数字的符号，而是值 231。由于这个额外的位，无符号整数的数值范围为 0 到 4294967295。对于小于 2147483647 的整数来说，无符号整数看来与有符号整数一样，而大于 2147483647 的整数则要使用位 31（在有符号整数中，这一位总是 0）。把无符号整数转换成字符串后，只返回它们的有效位。

注意：所有整数字面量都默认存储为有符号整数。只有 **ECMAScript** 的位运算符才能创建无符号整数。

位运算符

按位非 NOT

按位与 AND

按位或 OR

按位异或 XOR

左移运算 <<

有符号右移运算 >>

无符号右移运算 >>>

按位非运算符首先将数字转换成二进制(32位),然后再对各个数位上的1替换成0,而0则替换成1

```
var num = 12; /*12转换成二进制数为1100,32位形式为
   00000000000000000000000000001100     //取该二进制数的补码
   11111111111111111111111111110011    //-13 */ alert(~num);//-13,输出仍然以十进制格式
   //按位非实际上是对数字取负,然后再减一
```

按位与,按位或,按位异或运算符都是先将数字转换成32位二进制数,然后对各位进行操作

```
/* 0 & 0 == 0                    1 |  0 == 1
   0 & 1 == 0                    1 |  1 == 1
   1 & 1 == 1                    0 |  0 == 0
   异或比较特殊,当两加只有一个是1时返回1,其它返回0
   1 ^ 1 == 0                    1 ^ 0 == 1            0 ^ 0 == 0 */
   var num1 = 12; var num2 = 24; /* 00000000000000000000000000001100    // 12
   00000000000000000000000000011000    // 24
   -----------------------------------------------------    // &
   00000000000000000000000000001000    // 8 */ alert(num1 & num2);// 8
   /* 00000000000000000000000000001100    // 12
   00000000000000000000000000011000    // 24
   -----------------------------------------------------    // |
   00000000000000000000000000011100    // 28 */ alert(num1 |  num2);// 28
   /* 00000000000000000000000000001100    // 12
   00000000000000000000000000011000    // 24
   -----------------------------------------------------    // ^
   00000000000000000000000000010100    // 20 */ alert(num1 ^ num2);//20
   alert(num1 << 4);//将12左移4位,192
   /*左移运算符将二进制数字向左移动N位,空出的位置用0填充,超出32位的截去(不改变第32位)
   00000000000000000000000000001100
   00000000000000000000000011000000 //192 */ alert(num1 >> 4);//将12有符号右移4位,0
   /*有符号右移运算符将二进制数字向右移动N位,前面空出的位置用0填充(不改变第32位),后面截去N位
   00000000000000000000000000001100 //12
   00000000000000000000000000000000 //0 */ alert(num1 >>> 4);//将12无符号右移4位,0
   /*无符号右移运算符将二进制数字向右移动N位,前面空出的位置用0填充,后面截去N位
   //如果N大于0,第32位总会用0填充,使用无符号右移运算符得到的总是正数
   00000000000000000000000000001100 //12
   00000000000000000000000000000000 //0,对于正数,有符号右移和无符号右移一样 */ alert(-num1 >>>
   /* 11111111111111111111111111110100 //-12
   00001111111111111111111111111111 //268435455 */
```

Boolean 运算符

Boolean 运算符有三种：NOT(!)、AND(&&) 和 OR(||)。

逻辑 NOT 运算符的行为如下：

对于0,true,null,NaN,undefined,空字符串,返回true

对于非0数字,非空字符串,非空对象,true,返回false

```
//下面皆返回true
   alert(!null);
   alert(!0);
   alert(!undefined);
   alert(!false);
   alert(!"");
   alert(!NaN); //下面皆返回false
   alert(!" ");
   alert(!true);
   alert(!(new Object()));
   alert(!-1);
```

逻辑非运算符**(!)**始终返回布尔值**!**

当然,逻辑非运算符还有另外一个用途**---**〉将其它类型转换成布尔值，例如：

```
alert(!!NaN);//NaN的逻辑值为false
```

在 ECMAScript 中，逻辑 AND 运算符用双和号(&&)表示,逻辑OR运算符有双竖线表示(||),它们的效用你可能已经很清楚

```
alert(true && true);//true
alert(false && true);//false
alert(false && false);//false
alert(true || true);//true
alert(true || false);//true
alert(false || false);//false
```

然而在ECMAScript中，AND与OR运算符不但可以作用于逻辑值！

```
alert(34 && String);//将返回34
//但新手对这样的比较可能很迷惑
```

ECMAScript中的逻辑与逻辑或运算符又被称之为"快速与","快速或",或称之为"短路与","短路或".称之为短路或快速,是因为它们在测试两边条件时(将其转换成布尔值),如果第一个表达式的值已经能够确定整个表达式值的时候,就不会继续求下一个表达式的值,直接将这个值返回.比如与操作,如果第一个值转换为布尔是false,那么不管下个是true还是false,整个表达式的值都会是false,而对于OR运算符,则是如果第一个逻辑值为true,那么整个表达式的值就定了,就不用判断第二个条件了,所以,我们应该在AND运算中将最可能是false的条件放在前面,而在OR运算中,则将最有可能是true的条件放在前面,这样可以提高效率!我们可以用下面的例子来证明这一点:

```
function fnTrue() {
       alert(true); return true;
   } function fnFalse() {
       alert(false); return false;
   } //注意是执行两个函数,将返回值进行比较
   var a = fnFalse() && fnTrue();//只会出现一个弹窗,显示false
   //a = fnTrue() || fnFalse();//这一步只会出现一个弹窗,显示true
```

而对于这两个操作符的返回值,则只要记住一点:将最后判断的那个条件的值返回

```
//AND中第一个为true时,还要继续判断,第二个为false,整个表达式的值为false,同时将第二个条件返回
   alert(true && false); //当对其它类型使用AND时也是这样的
   alert(null && Object);//将返回null,因为在AND运算中第一条件为false,就无须继续了
   //同理,OR运算也是如此
   alert(Object || String);//返回Object,因为它的逻辑值是true
   alert(false || NaN);//返回NaN
```

数学运算符

加法(+),减法(-),乘法(*),除法(/)以及取模(%)都是小学数学的内容了,这里我们主要讨论在ECMAScript中,当这些运算符与一些特殊值相遇时会是什么情况

加法(+)

-Infinity 加 -Infinity，结果为 -Infinity。Infinity 加 Infinity，结果为 Infinity。

Infinity 加 -Infinity，结果为 NaN。

当只要有一边是字符串时,两边都被转换成字符串进行相连

减法(-)

-Infinity 减 Infinity，结果为 NaN。

-Infinity 减 -Infinity，结果为 NaN。

Infinity 减 -Infinity，结果为 Infinity。

-Infinity 减 Infinity，结果为 -Infinity。

乘法(*)

如果结果太大或太小，那么生成的结果是 Infinity 或 -Infinity。

Infinity 乘以 0 以外的任何数字，结果为 Infinity 或 -Infinity。

Infinity 乘以 Infinity，结果为 Infinity。

除法(/)

如果结果太大或太小，那么生成的结果是 Infinity 或 -Infinity。

Infinity 被 Infinity 除，结果为 NaN。

Infinity 被任何数字除，结果为 Infinity。

0 除一个任何非无穷大的数字，结果为 NaN。

Infinity 被 0 以外的任何数字除，结果为 Infinity 或 -Infinity。

取模(%)

如果被除数是 Infinity，或除数是 0，结果为 NaN。

Infinity 被 Infinity 除，结果为 NaN。

如果除数是无穷大的数，结果为被除数。

如果被除数为 0，结果为 0。

如果对其它类型的值进行数学运算,那么其它类型的值会自动转换成数字,如果转换失败,则返回 NaN(只有加法运算可以和普通字符串相连)

语句

判断语句:if (condition) {...} , if (condition) {....}else {....} , if (condition) {....} else if (condition) {......} else {.....}

循环语句:while (condition) {...} , do {...} while (condition) ,for (initialization; expression; post-loop-expression) statement; for (var key in obj) {......}

有标签的语句与break,continue语句

break 和 continue 语句的不同之处:break 语句可以立即退出循环，阻止再次反复执行任何代码。而 continue 语句只是退出当前循环，根据控制表达式还允许继续进行下一次循环。

```
var iNum = 0;
   outermost: for (var i=0; i<10; i++) { for (var j=0; j<10; j++) { if (i == 5 && j == 5
     }
     iNum++;
     }
   }
   alert(iNum); //输出 "55"
```

with 语句用于设置代码在特定对象中的作用域。

```
var sMessage = "hello"; with(sMessage) {
     alert(toUpperCase()); //输出 "HELLO"
} var person = new Object();
   person.age = 18;
   person.name = "XXX";
   person.parent = "ZZZ"; with (person) {
       alert(age+"\n"+name+"\n"+parent);
   }
```

with 语句是运行缓慢的代码块，尤其是在已设置了属性值时。大多数情况下，如果可能，最好避免使用它。

javascript快速入门11--正则表达式

正则表达式可以:

- 测试字符串的某个模式。例如，可以对一个输入字符串进行测试，看在该字符串是否存在一个电话号码模式或一个信用卡号码模式。这称为数据有效性验证
- 替换文本。可以在文档中使用一个正则表达式来标识特定文字，然后可以全部将其删除，或者替换为别的文字
- 根据模式匹配从字符串中提取一个子字符串。可以用来在文本或输入字段中查找特定文字

正则表达式语法

一个正则表达式就是由普通字符（例如字符 a 到 z）以及特殊字符（称为元字符）组成的文字模式。该模式描述在查找文字主体时待匹配的一个或多个字符串。正则表达式作为一个模板，将某个字符模式与所搜索的字符串进行匹配。

下表是元字符及其在正则表达式上下文中的行为的一个完整参考列表：

字符	描述
\	转义字符,在之前,我们在字符串也用过这字符,即一些字符具有特殊含义,对其进行转义使它成为普通字符,而用在普通字符上又表示其有特殊含义
^	匹配输入字符串的开始位置,如果设置了匹配多行(m),那么也匹配行的开头
$	匹配输入字符串的结束位置。如果设置了匹配多行(m),那么也匹配行的结束
*	匹配前面的子表达式零次或多次。例如，zo 能匹配 *"z"* 以及 *"zoo"*。 等价于{0,}。
+	匹配前面的子表达式一次或多次。例如，'zo+' 能匹配 "zo" 以及 "zoo"，但不能匹配 "z"。+ 等价于 {1,}。
?	匹配前面的子表达式零次或一次。例如，"do(es)?" 可以匹配 "do" 或 "does" 中的"do" 。? 等价于 {0,1}。
{*n*}	*n* 是一个非负整数。匹配确定的 *n* 次。例如，'o{2}' 不能匹配 "Bob" 中的 'o'，但是能匹配 "food" 中的两个 o。
{*n*,}	*n* 是一个非负整数。至少匹配*n* 次。例如，'o{2,}' 不能匹配 "Bob" 中的 'o'，但能匹配 "foooood" 中的所有 o。'o{1,}' 等价于 'o+'。'o{0,}' 则等价于 'o*'。
{*n*,*m*}	*m* 和 *n* 均为非负整数，其中*n* <= *m*。最少匹配 *n* 次且最多匹配 *m* 次。刘，"o{1,3}" 将匹配 "fooooood" 中的前三个 o。'o{0,1}' 等价于 'o?'。请注意在逗号和两个数之间不能有空格。
	当该字符紧跟在任何一个其他限制符 (*, +, ?, {*n*}, {*n*,}, {*n*,*m*}) 后面时，匹配

<table>
<tr><td>?</td><td>模式是非贪婪的。非贪婪模式尽可能少的匹配所搜索的字符串，而默认的贪婪模式则尽可能多的匹配所搜索的字符串。例如，对于字符串 "oooo"，'o+?' 将匹配单个 "o"，而 'o+' 将匹配所有 'o'。</td></tr>
<tr><td>.</td><td>匹配除 "\n" 之外的任何单个字符。要匹配包括 '\n' 在内的任何字符，请使用象 '[.\n]' 的模式。</td></tr>
<tr><td>(pattern)</td><td>匹配pattern 并获取这一匹配。所获取的匹配可以从产生的 Matches 集合得到，使用 $0…$9 属性。要匹配圆括号字符，请使用 '(' 或 ')'。</td></tr>
<tr><td>(?:pattern)</td><td>匹配 pattern 但不获取匹配结果，也就是说这是一个非获取匹配，不进行存储供以后使用。这在使用 "或" 字符 (|) 来组合一个模式的各个部分是很有用。例如， 'industr(?:y|ies) 就是一个比 'industry|industries' 更简略的表达式。</td></tr>
<tr><td>(?=pattern)</td><td>正向预查，在任何匹配 pattern 的字符串开始处匹配查找字符串。这是一个非获取匹配，也就是说，该匹配不需要获取供以后使用。例如， 'Windows (?=95|98|NT|2000)' 能匹配 "Windows 2000" 中的 "Windows" ，但不能匹配 "Windows 3.1" 中的 "Windows"。预查不消耗字符，也就是说，在一个匹配发生后，在最后一次匹配之后立即开始下一次匹配的搜索，而不是从包含预查的字符之后开始。</td></tr>
<tr><td>(?!pattern)</td><td>负向预查，在任何不匹配pattern 的字符串开始处匹配查找字符串。这是一个非获取匹配，也就是说，该匹配不需要获取供以后使用。例如'Windows (?!95|98|NT|2000)' 能匹配 "Windows 3.1" 中的 "Windows"，但不能匹配 "Windows 2000" 中的 "Windows"。预查不消耗字符，也就是说，在一个匹配发生后，在最后一次匹配之后立即开始下一次匹配的搜索，而不是从包含预查的字符之后开始</td></tr>
<tr><td>x|y</td><td>匹配 x 或 y。例如，'z|food' 能匹配 "z" 或 "food"。'(z|f)ood' 则匹配 "zood" 或 "food"。</td></tr>
<tr><td>[xyz]</td><td>字符集合。匹配所包含的任意一个字符。例如， '[abc]' 可以匹配 "plain" 中的 'a'。</td></tr>
<tr><td>[^xyz]</td><td>负值字符集合。匹配未包含的任意字符。例如， 'abc' 可以匹配 "plain" 中的'p'。</td></tr>
<tr><td>[a-z]</td><td>字符范围。匹配指定范围内的任意字符。例如，'[a-z]' 可以匹配 'a' 到 'z' 范围内的任意小写字母字符。</td></tr>
<tr><td>[^a-z]</td><td>负值字符范围。匹配任何不在指定范围内的任意字符。例如，'a-z' 可以匹配任何不在 'a' 到 'z' 范围内的任意字符。</td></tr>
<tr><td>\b</td><td>匹配一个单词边界，也就是指单词和空格间的位置。例如， 'er\b' 可以匹配"never" 中的 'er'，但不能匹配 "verb" 中的 'er'。</td></tr>
<tr><td>\B</td><td>匹配非单词边界。'er\B' 能匹配 "verb" 中的 'er'，但不能匹配 "never" 中的 'er'。</td></tr>
<tr><td>\cx</td><td>匹配由x指明的控制字符。例如， \cM 匹配一个 Control-M 或回车符。 x 的值必须为 A-Z 或 a-z 之一。否则，将 c 视为一个原义的 'c' 字符。</td></tr>
<tr><td>\d</td><td>匹配一个数字字符。等价于 [0-9]。</td></tr>
<tr><td>\D</td><td>匹配一个非数字字符。等价于 0-9 。</td></tr>
<tr><td></td><td></td></tr>
</table>

\f	匹配一个换页符。等价于 \x0c 和 \cL。
\n	匹配一个换行符。等价于 \x0a 和 \cJ。
\r	匹配一个回车符。等价于 \x0d 和 \cM。
\s	匹配任何空白字符，包括空格、制表符、换页符等等。等价于 [?\f\n\r\t\v]。
\S	匹配任何非空白字符。等价于 ?\f\n\r\t\v 。
\t	匹配一个制表符。等价于 \x09 和 \cI。
\v	匹配一个垂直制表符。等价于 \x0b 和 \cK。
\w	匹配包括下划线的任何单词字符。等价于'[A-Za-z0-9_]'。
\W	匹配任何非单词字符。等价于 'A-Za-z0-9_' 。
\x*n*	匹配 *n*，其中 *n* 为十六进制转义值。十六进制转义值必须为确定的两个数字长。例如， '\x41' 匹配 "A"。'\x041' 则等价于 '\x04' & "1"。正则表达式中可以使用 ASCII 编码。.
num	匹配 *num*，其中 *num* 是一个正整数。对所获取的匹配的引用。例如，'(.)\1' 匹配两个连续的相同字符。
_n_	标识一个八进制转义值或一个后向引用。如果 _n 之前至少 _*n* 个获取的子表达式，则 *n* 为后向引用。否则，如果 *n* 为八进制数字 (0-7)，则 *n* 为一个八进制转义值。
_nm_	标识一个八进制转义值或一个后向引用。如果 _nm 之前至少有*is preceded by at least* _*nm* 个获取得子表达式，则 *nm* 为后向引用。如果 _nm 之前至少有 _*n* 个获取，则 *n* 为一个后跟文字 *m* 的后向引用。如果前面的条件都不满足，若? *n* 和 *m* 均为八进制数字 (0-7)，则 _nm 将匹配八进制转义值 _*nm*。
nml	如果 *n* 为八进制数字 (0-3)，且 *m* 和 *l* 均为八进制数字 (0-7)，则匹配八进制转义值 *nml*。
\u*n*	匹配 *n*，其中 *n* 是一个用四个十六进制数字表示的 Unicode 字符。例如，\u00A9 匹配版权符号 (?)。

创建正则表达式

```
var re = new RegExp();//RegExp是一个对象,和Aarray一样
   //但这样没有任何效果,需要将正则表达式的内容作为字符串传递进去
   re =new RegExp("a");//最简单的正则表达式,将匹配字母a
   re=new RegExp("a","i");//第二个参数,表示匹配时不分大小写
```

RegExp构造函数第一个参数为正则表达式的文本内容,而第一个参数则为可选项标志.标志可以组合使用

- g （全文查找）
- i （忽略大小写）

- m （多行查找）

```
var re = new RegExp("a","gi");//匹配所有的a或A
```

正则表达式还有另一种正则表达式字面量的声明方式

```
var re = /a/gi;
```

和正则表达式相关的方法和属性

正则表达式对象的方法

- test,返回一个 Boolean 值，它指出在被查找的字符串中是否存在模式。如果存在则返回 true，否则就返回 false。
- exec,用正则表达式模式在字符串中运行查找，并返回包含该查找结果的一个数组。
- compile,把正则表达式编译为内部格式，从而执行得更快。

正则表达式对象的属性

- source,返回正则表达式模式的文本的复本。只读。
- lastIndex,返回字符位置，它是被查找字符串中下一次成功匹配的开始位置。
- $1...$9,返回九个在模式匹配期间找到的、最近保存的部分。只读。
- input ($_),返回执行规范表述查找的字符串。只读。
- lastMatch ($&),返回任何正则表达式搜索过程中的最后匹配的字符。只读。
- lastParen ($+),如果有的话，返回任何正则表达式查找过程中最后括的子匹配。只读。
- leftContext ($`),返回被查找的字符串中从字符串开始位置到最后匹配之前的位置之间的字符。只读。
- rightContext ($'),返回被搜索的字符串中从最后一个匹配位置开始到字符串结尾之间的字符。只读。

String对象一些和正则表达式相关的方法

- match,找到一个或多个正则表达式的匹配。
- replace,替换与正则表达式匹配的子串。
- search,检索与正则表达式相匹配的值。
- split,把字符串分割为字符串数组。

测试正则表达式是如何工作的!

```
//test方法,测试字符串,符合模式时返回true,否则返回false
   var re = /he/;//最简单的正则表达式,将匹配he这个单词
   var str = "he";
   alert(re.test(str));//true
   str = "we";
   alert(re.test(str));//false
   str = "HE";
   alert(re.test(str));//false,大写,如果要大小写都匹配可以指定i标志(i是ignoreCase或case-insens
   re = /he/i;
   alert(re.test(str));//true
   str = "Certainly!He loves her!";
   alert(re.test(str));//true,只要包含he(HE)就符合,如果要只是he或HE,不能有其它字符,则可使用^和$
   re = /^he/i;//脱字符(^)代表字符开始位置
   alert(re.test(str));//false,因为he不在str最开始
   str = "He is a good boy!";
   alert(re.test(str));//true,He是字符开始位置,还需要使用$
   re = /^he$/i;//$表示字符结束位置
   alert(re.test(str));//false
   str = "He";
   alert(re.test(str));//true
   //当然,这样不能发现正则表达式有多强大,因为我们完全可以在上面的例子中使用==或indexOf
   re = /\s/;// \s匹配任何空白字符，包括空格、制表符、换页符等等
   str= "user Name";//用户名包含空格
   alert(re.test(str));//true
   str = "user        Name";//用户名包含制表符
   alert(re.test(str));//true
   re=/^[a-z]/i;//[]匹配指定范围内的任意字符,这里将匹配英文字母,不区分大小写
   str="variableName";//变量名必须以字母开头
   alert(re.test(str));//true
   str="123abc";
   alert(re.test(str));//false
```

当然,仅仅知道了字符串是否匹配模式还不够,我们还需要知道哪些字符匹配了模式

```
var osVersion = "Ubuntu 8";//其中的8表示系统主版本号
   var re = /^[a-z]+\s+\d+$/i; //+号表示字符至少要出现1次,\s表示空白字符,\d表示一个数字
   alert(re.test(osVersion));//true,但我们想知道主版本号
   //另一个方法exec,返回一个数组,数组的第一个元素为完整的匹配内容
   re=/^[a-z]+\s+\d+$/i;
   arr = re.exec(osVersion);
   alert(arr[0]);//将osVersion完整输出,因为整个字符串刚好匹配re
   //我只需要取出数字
   re=/\d+/; var arr = re.exec(osVersion);
   alert(arr[0]);//8
```

更复杂的用法,使用子匹配

```
//exec返回的数组第1到n元素中包含的是匹配中出现的任意一个子匹配
   re=/^[a-z]+\s+(\d+)$/i;//用()来创建子匹配
   arr =re.exec(osVersion);
   alert(arr[0]);//整个osVersion,也就是正则表达式的完整匹配
   alert(arr[1]);//8,第一个子匹配,事实也可以这样取出主版本号
   alert(arr.length);//2
   osVersion = "Ubuntu 8.10";//取出主版本号和次版本号
   re = /^[a-z]+\s+(\d+)\.(\d+)$/i;//.是正则表达式元字符之一,若要用它的字面意义须转义
   arr = re.exec(osVersion);
   alert(arr[0]);//完整的osVersion
   alert(arr[1]);//8
   alert(arr[2]);//10
```

注意,当字符串不匹配**re**时,**exec**方法将返回**null**

String对象的一些和正则表达式有关的方法

```
//replace方法,用于替换字符串
   var str ="some money";
   alert(str.replace("some","much"));//much money
   //replace的第一个参数可以为正则表达式
   var re = /\s/;//空白字符
   alert(str.replace(re,"%"));//some%money
   //在不知道字符串中有多少空白字符时,正则表达式极为方便
   str ="some some             \tsome\t\f";
   re = /\s+/;
   alert(str.replace(re,"#"));//但这样只会将第一次出现的一堆空白字符替换掉
   //因为一个正则表达式只能进行一次匹配,\s+匹配了第一个空格后就退出了
   re = /\s+/g;//g,全局标志,将使正则表达式匹配整个字符串
   alert(str.replace(re,"@"));//some@some@some@
   //另一个与之相似的是split
   var str = "a-bd-c"; var arr = str.split("-");//返回["a","bd","c"]
   //如果str是用户输入的,他可能输入a-bd-c也可能输入a bd c或a_bd_c,但不会是abdc(这样就说他输错了)
   str = "a_db-c";//用户以他喜欢的方式加分隔符s
   re=/[^a-z]/i;//前面我们说^表示字符开始,但在[]里它表示一个负字符集
   //匹配任何不在指定范围内的任意字符,这里将匹配除字母处的所有字符
   arr = str.split(re);//仍返回["a","bd","c"];
   //在字符串中查找时我们常用indexOf,与之对应用于正则查找的方法是search
   str = "My age is 18.Golden age!";//年龄不是一定的,我们用indexOf不能查找它的位置
   re = /\d+/;
   alert(str.search(re));//返回查找到的字符串开始下标10
   //注意,因为查找本身就是出现第一次就立即返回,所以无需在search时使用g标志
   //下面的代码虽然不出错,但g标志是多余的
   re=/\d+/g;
   alert(str.search(re));//仍然是10
```

注意,当**search**方法没有找到匹配时,将返回**-1**

类似于exec方法,String对象的match方法也用于将字符串与正则表达式进行匹配并返回结果数组

```
var str = "My name is CJ.Hello everyone!"; var re = /[A-Z]/;//匹配所有大写字母
   var arr = str.match(re);//返回数组
   alert(arr);//数组中只会包含一个M,因为我们没有使用全局匹配
   re = /[A-Z]/g;
   arr = str.match(re);
   alert(arr);//M,C,J,H
   //从字符串中抽取单词
   re = /\b[a-z]\b/i;//\b表示单词边界
   str = "one two three four";
   alert(str.match(re));//one,two,three,four
```

RegExp对象实例的一些属性

```
var re = /[a-z]/i;
   alert(re.source);//将[a-z]字符串输出
   //请注意,直接alert(re)会将正则表达式连同前向斜线与标志输出,这是re.toString方法定义的
```

每个RegExp对象的实例具有lastIndex属性,它是被查找字符串中下一次成功匹配的开始位置,默认值是-1。 lastIndex 属性被 RegExp 对象的 exec 和 test 方法修改.并且它是可写的.

```
var re = /[A-Z]/; //exec方法执行后,修改了re的lastIndex属性,
   var str = "Hello,World!!!"; var arr = re.exec(str);
   alert(re.lastIndex);//0,因为没有设置全局标志
   re = /[A-Z]/g;
   arr = re.exec(str);
   alert(re.lastIndex);//1
   arr = re.exec(str);
   alert(re.lastIndex);//7
```

当匹配失败（后面没有匹配），或lastIndex值大于字符串长度时，再执行exec等方法会将lastIndex设为0(开始位置)

```
var re = /[A-Z]/; var str = "Hello,World!!!";
   re.lastIndex = 120; var arr = re.exec(str);
   alert(re.lastIndex);//0
```

RegExp对象的静态属性

```
//input 最后用于匹配的字符串（传递给test,exec方法的字符串）
   var re = /[A-Z]/; var str = "Hello,World!!!"; var arr = re.exec(str);
   alert(RegExp.input);//Hello,World!!!
   re.exec("tempstr");
   alert(RegExp.input);//仍然是Hello,World!!!,因为tempstr不匹配
   //lastMatch 最后匹配的字符
   re = /[a-z]/g;
   str = "hi";
   re.test(str);
   alert(RegExp.lastMatch);//h
re.test(str);
   alert(RegExp["$&"]);//i  ,$&是lastMatch的短名字，但由于它不是合法变量名，所以要。。
   //lastParen 最后匹配的分组
   re = /[a-z](\d+)/gi;
   str = "Class1 Class2 Class3";
   re.test(str);
   alert(RegExp.lastParen);//1
re.test(str);
   alert(RegExp["$+"]);//2
   //leftContext  返回被查找的字符串中从字符串开始位置到最后匹配之前的位置之间的字符
   //rigthContext 返回被搜索的字符串中从最后一个匹配位置开始到字符串结尾之间的字符
   re = /[A-Z]/g;
   str = "123ABC456";
   re.test(str);
   alert(RegExp.leftContext);//123
   alert(RegExp.rightContext);//BC456
re.test(str);
   alert(RegExp["$`"]);//123A
   alert(RegExp["$'"]);//C456
```

multiline属性返回正则表达式是否使用多行模式,这个属性不针对某个正则表达式实例，而是针对所有正则表达式，并且这个属性可写.(IE与Opera不支持这个属性)

```
alert(RegExp.multiline); //因为IE，Opera不支持这个属性，所以最好还是单独指定
   var re = /\w+/m;
   alert(re.multiline);
   alert(RegExp["$*"]);//RegExp对象的静态属性不会因为给RegExp某个对象实例指定了m标志而改变
   RegExp.multiline = true;//这将打开所有正则表达式实例的多行匹配模式
   alert(RegExp.multiline);
```

正则表达式高级篇

使用元字符注意事项:元字符是正则表达式的一部分，当我们要匹配正则表达式本身时，必须对这些元字符转义.下面是正则表达式用到的所有元字符

```
( [ { \ ^ $ | ) ? * + .
```

```
var str = "?"; var re = /?/;
   alert(re.test(str));//出错，因为？是元字符，必须转义
   re = /\?/;
   alert(re.test(str));//true
```

使用RegExp构造函数与使用正则表达式字面量创建正则表达式注意点

```
var str = "\?";
   alert(str);//只会输出?
   var re = /\?/;//将匹配?
   alert(re.test(str));//true
   re = new RegExp("\?");//出错,因为这相当于re = /\?/
   re = new RegExp("\\?");//正确，将匹配？
   alert(re.test(str));//true
```

既然双重转义这么不友好，所以还是用正则表达式字面量的声明方式

如何在正则表达式中使用特殊字符？

```
//ASCII方式用十六进制数来表示特殊字符
   var re = /^\x43\x4A$/;//将匹配CJ
   alert(re.test("CJ"));//true
   //也可使用八进制方式
   re = /^\103\112$/;//将匹配CJ
   alert(re.test("CJ"));//true
   //还可以使用Unicode编码
   re =/^\u0043\u004A$/;//使用 Unicode，必须使用u开头，接着是字符编码的四位16进制表现形式
   alert(re.test("CJ"));
```

另处，还有一些其它的预定义特殊字符，如下表所示:

字符	描述
\n	换行符
\r	回车符
\t	制表符
\f	换页符（Tab）
\cX	与X对应的控制字符
\b	退格符(BackSpace)
\v	垂直制表符
\0	空字符("")

字符类 ---〉简单类，反向类，范围类，组合类，预定义类

```
//简单类
    var re = /[abc123]/;//将匹配abc123这6个字符中一个
    //负向类
    re = /[^abc]/;//将匹配除abc之外的一个字符
    //范围类
    re = /[a-b]/;//将匹配小写a-b 26个字母
    re = /[^0-9]/;//将匹配除0-9 10个字符之处的一个字符
    //组合类
    re = /[a-b0-9A-Z_]/;//将匹配字母，数字和下划线
```

下面是正则表达式中的预定义类

代码	等同于	匹配
.	IE下\n，其它\n\r	匹配除换行符之外的任何一个字符
\d	[0-9]	匹配数字
\D	0-9	匹配非数字字符
\s	[\n\r\t\f\x0B]	匹配一个空白字符
\S	\n\r\t\f\x0B	匹配一个非空白字符
\w	[a-zA-Z0-9_]	匹配字母数字和下划线
\W	a-zA-Z0-9_	匹配除字母数字下划线之外的字符

量词

代码	描述
*	匹配前面的子表达式零次或多次。例如，zo 能匹配 *"z"* 以及 *"zoo"*。 等价于{0,}。
+	匹配前面的子表达式一次或多次。例如，'zo+' 能匹配 "zo" 以及 "zoo"，但不能匹配 "z"。+ 等价于 {1,}。
?	匹配前面的子表达式零次或一次。例如，"do(es)?" 可以匹配 "do" 或 "does" 中的"do" 。? 等价于 {0,1}。
{*n*}	*n* 是一个非负整数。匹配确定的 *n* 次。例如，'o{2}' 不能匹配 "Bob" 中的 'o'，但是能匹配 "food" 中的两个 o。
{*n*,}	*n* 是一个非负整数。至少匹配*n* 次。例如，'o{2,}' 不能匹配 "Bob" 中的 'o'，但能匹配 "foooood" 中的所有 o。'o{1,}' 等价于 'o+'。'o{0,}' 则等价于 'o*'。
{*n*,*m*}	*m* 和 *n* 均为非负整数，其中*n* <= *m*。最少匹配 *n* 次且最多匹配 *m* 次。刘，"o{1,3}" 将匹配 "fooooood" 中的前三个 o。'o{0,1}' 等价于 'o?'。请注意在逗号和两个数之间不能有空格。

贪婪量词与惰性量词

- 用贪婪量词进行匹配时，它首先会将整会字符串当成一个匹配，如果匹配的话就退出，如果不匹配，就截去最后一个字符进行匹配，如果不匹配，继续将最后一个字符截去进行匹配，直到有匹配为止。直到现在我们遇到的量词都是贪婪量词
- 用贪婪量词进行匹配时，它首先将第一个字符当成一个匹配，如果成功则退出，如果失败，则测试前两个字符，依些增加，直到遇到合适的匹配为止

贪婪	惰性
+	+?
?	??
*	*?
{n}	{n}?
{n,}	{n,}?
{n,m}	{n,m}

```
var str = "abc"; var re = /\w+/;//将匹配abc
   re = /\w+?/;//将匹配a
```

多行模式

```
var re = /[a-z]$/; var str = "ab\ncdef";
   alert(str.replace(re,"#"));//ab\ncde#
   re =/[a-z]$/m;
   alert(str.replace(re,"#"));//a#\ncde#
```

分组与非捕获性分组

```
re = /abc{2}/;//将匹配abcc
re = /(abc){2}/;//将匹配abcabc
//上面的分组都是捕获性分组
str = "abcabc ###";
arr = re.exec(str);
alert(arr[1]);//abc
//非捕获性分组 (?:)
re = /(?:abc){2}/;
arr = re.exec(str);
alert(arr[1]);//undefined
```

候选（也就是所说的"或"）

```
re = /^a|bc$/;//将匹配开始位置的a或结束位置的bc
str ="add";
alert(re.test(str));//true
re = /^(a|bc)$/;//将匹配a或bc
str ="bc";
alert(re.test(str));//true
```

当包含分组的正则表达式进行过test,match,search这些方法之后，每个分组都被放在一个特殊的地方以备将来使用，这些存储是分组中的特殊值，我们称之为反向引用

```
var re = /(A?(B?(C?)))/; /*上面的正则表达式将依次产生三个分组
   (A?(B?(C?))) 最外面的
   (B?(C?))
   (C?)*/ str = "ABC";
   re.test(str);//反向引用被存储在RegExp对象的静态属性$1—$9中
   alert(RegExp.$1+"\n"+RegExp.$2+"\n"+RegExp.$3); //反向引用也可以在正则表达式中使用\1 ,\2..
   re = /\d+(\D)\d+\1\d+/;
   str = "2008-1-1";
   alert(re.test(str));//true
   str = "2008-4_3";
   alert(re.test(str));//false
```

使用反向引用可以要求字符串中某几个位置上的字符必须相同.另外，在replace这类方法可，存在特殊字符序列来表示反向引用的方式

```
re = /(\d)\s(\d)/;
str = "1234 5678";
alert(str.replace(re,"$2 $1"));//在这个里面$1表示第一个分组1234,$2则表示5678
```

其它——〉正向前瞻,用来捕获出现在特定字符之前的字符,只有当字符后面跟着某个特定字符才去捕获它。与正向前瞻对应的有负向前瞻，它用匹配只有当字符后面不跟着某个特定字符时才去匹配它。在执行前瞻和负向前瞻之类的运算时，正则表达式引擎会留意字符串后面的部分，然而却不移动index

```
//正向前瞻
    re = /([a-z]+(?=\d))/i; //我们要匹配后面跟一个数字的单词，然后将单词返回，而不要返回数字
    str = "abc every1 abc";
    alert(re.test(str));//true
    alert(RegExp.$1);//every
    alert(re.lastIndex);//使用前瞻的好处是,前瞻的内容(?=\d)并不会当成一次匹配，下次匹配仍从它开始
    //负向前瞻(?!)
    re = /([a-z](?!\d))/;i //将匹配后面不包含数字的字母,并且不会返回(?!\d)中的内容
    str = "abc1 one";
    alert(re.test(str));
    alert(RegExp.$1);//one
```

构建一个验证电子邮箱地址有效性的正则表达式。电子邮箱地址有效性要求(我们姑且这样定义)：用户名只能包含字母数字以及下划线，最少一位，最多25位，用户名后面紧跟@，后面是域名，域名名称要求只能包含字母数字和减号（-），并且不能以减号开头或结尾，然后后面是域名后缀（可以有多个），域名后缀必须是点号连上2-4位英文字母

```
var re = /^\w{1,15}(?:@(?!-))(?:(?:[a-z0-9-]*)(?:[a-z0-9](?!-))(?:\.(?!-)))+[a-z]{2,4}$/
```

javascript快速入门12--函数式与面向对象

函数

函数是一组可以随时随地运行的语句。

函数是 ECMAScript 的核心。

创建函数

```
function fnOne() {//具有名称的函数,函数名必须符合变量名命名规范
       //可以没有符何语句
}

   var fnTwo = function () {//匿名函数
};

   function () {//创建匿名函数而不立即创建其引用，那么之后就没办法调用此函数
}

   (function fnThree() {
   })();//创建函数并立即执行一次

   (function () {})();//创建匿名函数并立即执行一次
```

匿名函数与命名函数的区别

```
   fnOne();//不会出错，使用function创建的具有名称的函数在任何与之相同作用域的地方都能调用
   fnTwo();//出错
   var fnTwo =function () {};//因为只有执行了这个赋值语句后，fnTwo才会被创建
   function fnOne() {}
```

函数返回值

```
function fnTest() { return "value";//使用return来返回值
       alert("Hello!!!");//执行了return之后，函数就会退出
   }
```

函数参数

```
function fnTest(arg1,arg2,arg3) {
       alert(arg1+"\n"+arg2+"\n"+arg3);
   }
   fnTest(1,2,3);
   fnTest(1,2);//没有传值过去时，就会是undefined
```

arguments对象:在函数执行时函数内部就会有arguments对象，它包含了所有的参数，arguments的length属性报告了传入参数个数

```
function fnTest() { for (var i=0;i< arguments.length;i++) {
           alert(arguments[i]);
       }
   }
   fnTest(1,2,3);
   fnTest(45);
```

使用arguments对象模拟函数重载

```
function fnTest() { var args = arguments; switch (arguments.length) { case 0 : return "
       }
   }
   alert(fnTest());
   alert(fnTest(1));
   alert(fnTest(2));
```

arguments对象补充:arguments对象的callee属性指向它所在的函数

```
function fnTest() {alert(arguments.callee);}
```

闭包

闭包，指的是词法表示包括不被计算的变量的函数，也就是说，函数可以使用函数之外定义的变量。

在 ECMAScript 中使用全局变量是一个简单的闭包实例。请思考下面这段代码：

```
var msg = "我是全局变量!!!"; function say() {
     alert(msg);
   }
   say();
```

在ECMAScript中，在函数声明处向函数外部看到的声明的所有变量，在函数内部都能访问到它们的最终值！

```
var g = "全局变量！！！"; function fnA() { var a="A"; function fnB() { var b="B";
           alert(a);//可以访问到a
           alert(c);//但不以访问c
           function fnC() { var c = "C";
               alert(a+"\n"+b);//只要遵循从里向外看的原则，看到的变量都可以访问到
}
       }
   } //更复杂的闭包
   function fnTest(num1,num2) { var num3 = num1+num2; return function () {
           alert("num1+num2结果为"+num3);
       };
   } var result = fnTest(23,56);
   result();
```

闭包函数只能访问变量的最终值!!!

```
function fnTest(arr) { for (var i=0;i < arr.length;i++) {
            arr[i]=function () {
                alert(i+" | "+arr[i]);
            };
        }
    } var arr = [0,1,2,3,4,5];
    fnTest(arr); for (var i=0;i < arr.length;i++) {
        arr[i]();//始终输出6还有一个undefined
        //因为函数退出后，i值为6,所以访问到的值只有6
    }
```

不但在闭包中可以访问闭包外的变量值，而且还可以设置它的值

```
function fnTest() { var a=123; return {
            set:function (param) {a = param},
            get:function () {return a}
        };
    } var obj = fnTest();
    alert(obj.get());//123
    obj.set(4);
    alert(obj.get());//4
```

对象,构造函数

创建一个对象

```
var obj = new Object();
    alert(obj);
    alert(Object);//一个函数
    Object();//可以直接执行
    //构造函数也是一个普通的函数
    function Demo() {
    } var d = new Demo();//不会出错，使用new运算符来创建对象实例
    alert(d);//object
```

this关键字的用法

```
function Demo() { this.property = "属性！！！";
    }
    d = new Demo();
    alert(d.property);//属性！！！
```

不使用new而直接执行构造函数时，this指向window

```
    function Demo() { this.property = "属性！！！";
    } var d = Demo();
    alert(d.property);//undefined
    alert(window.property);//属性！！！
```

可以给构造函数传递参数,然后可以将参数赋值给对象的属性

```
function Person(name,age) { this.name = name; this.age = age;
   } var p1 = new Person("CJ",18);
```

instanceof 运算符，用来判断对象是否是某个类（虽然ECMAScript中并不存在类，但我们在这里依然使用这一术语）的实例

```
var str = new String("string");
   alert(str instanceof String);//true
   var arr = new Array();
   alert(arr instanceof Array);//true
   function Demo() {} var d = new Demo();
   alert(d instanceof Demo);//true
```

面向对象术语

一种面向对象语言需要向开发者提供四种基本能力：

- 封装——把相关的信息（无论数据或方法）存储在对象中的能力。
- 聚集——把一个对象存储在另一个对象内的能力。
- 继承——由另一个类（或多个类）得来类的属性和方法的能力。
- 多态——编写能以多种方法运行的函数或方法的能力。

ECMAScript支持这些要求，因此可被看作面向对象的。

封装与私有属性:封装并不要求私有！

```
function Person(name,age) { this.name = name;//将值存储为对象的属性即是封装
       this.age = age;
   } var p1 = new Person("CJ",18);
```

ECMAScript目前版本并不支持私有属性，但可以通过闭包来模拟

```
function Person(name,age) { this.getName = function () {return name}; this.setName = fun
   } var p1 = new Person("CJ",18);
   alert(p1.name);//undefined
   alert(p1.getName());//CJ
   p1.setName("XXX");
   alert(p1.getName());//XXX
```

继承：prototype属性

ECMAScript中，继承是通过构造函数的prototype属性实现的

```
function Person(name,age) { this.name=name; this.age = name;
   }
   alert(Person.prototype);//object
   Person.prototype.sort = "人"; var p1 = new Person("CJ",18);
   alert(p1.sort);//所有的Person对象的实例继承了sort这个属性
```

所有对象都有一个方法isPrototypeOf(),用来判断它是不是另一个对象的原型

```
function Person() {} var p1 = new Person();
   alert(Person.prototype.isPrototypeOf(p1));//true
```

在ECMAScript中让一个类继承另一个类的方式比较特殊

```
function ClassA() { this.a = "A";
   } function ClassB() { this.b = "B";
   }
   ClassB.prototype = new ClassA(); //让ClassB继承ClassA
   var b = new ClassB();
   alert(b.a);//"A",继承了属性a
   alert(b instanceof ClassB);//true
   alert(b instanceof ClassA);//true,因为继承，b也是ClassA的后代
   alert(ClassB.prototype.isPrototypeOf(b));//true
   alert(ClassA.prototype.isPrototypeOf(b));//true,ClassA.prototype也是b的原型
```

然而这样的继承有个注意点——

```
function ClassA() { this.a = "A";
   } function ClassB() { this.b = "B";
   } var b = new ClassB();//先实例化ClassB
   ClassB.prototype = new ClassA();//再去继承ClassA,将prototype重置为另一个对象
   alert(b instanceof ClassB);//false
   alert(b instanceof ClassA);//false
   alert(ClassB.prototype.isPrototypeOf(b));//false
   alert(ClassA.prototype.isprototypeOf(b));//false
```

当构造函数需要参数时

```
function Person(name,age) { this.name = name; this.age = age;
   } function Teacher(name,age,lesson) { this.tempMethod = Person;//对象冒充
       this.tempMethod(name,age); //当执行Person时,由于是在Teacher某个实例上调用的,所以在Perso
       delete this.tempMethod;//删除临时方法
       this.lesson = lesson;
   }
   ClassB.prototype =new ClassA();//始终不应在继承时放参数
   var t1 = new Teacher("HUXP",18,"C#");
   alert(t1.name+" | "+ this.age+ " | "+this.lesson);
```

事实上，对于对象冒充，ECMAScript提供了更简洁的内置方法call,在每个函数上调用，第一个参数即为要冒充的对象，剩下的是函数需要的其它参数

```
function Demo(arg) { this.name = arg;
   } var obj = new Object();
   Demo.call(obj,"name");
   alert(obj.name);//"name"
   //使用call重写上面继承的例子
   function Person(name,age) { this.name = name; this.age = age;
   } function Teacher(name,age,lesson) {
       Person.call(this,name,age);//对象冒充
       this.lesson = lesson;
   }
```

静态属性与Function类

在ECMAScript里有个有趣的地方是，函数本身也是对象（和数组也一样），也可使用new来创建.Function构造函数至少要传两个字符串参数，可以是空字符串。除了最后一个字符串会被当做函数体语句去执行，其它参数都会作为函数参数变量名!

```
var fn = new Function();//一个空函数
   //创建有内容的函数
   fn = new Function("arg1","alert(arg1)");//最后一个参数为执行语句的字符串，前面参数全是函数要用
   //上面的代码等效于
   fn = function (arg1) {alert(arg1)};//同样，由于都是赋值语句，所以要注意出现次序
   fn.property ="既然是对象，那么就要以添加属性"
```

事实上，在构造函数上添加的属性就被称之为静态属性，它不会被实例所继承

```
function ClassDemo() {
   }
   ClassDemo.property = new Array(); var d =new ClassDemo();
   alert(d.property);//undefined
   alert(ClassDemo.property);//object
```

Function类的实例还有其它一些方法和属性（当然，使用function关键字声明的函数也是一样的）

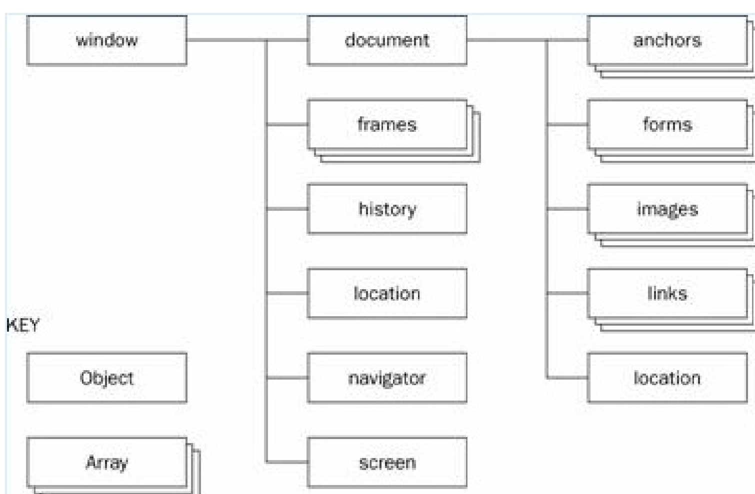
```
function outer() {
       inner(); function inner() {
           alert(inner.caller); //函数的caller属性指向调用自身的上下文函数，这里指向outer
}
   } function applyTest() { var args = arguments; this.name = args[0]; this.age=args[1];
   } var obj = new Object();
   applyTest.apply(obj,["name",18]);
   alert(obj.name);
   alert(obj.age); //apply方法与call方法类似，也用于对象冒充，只是apply方法只有两个参数
   //第一个是要冒充的对象，第二个是所有参数组成的数组
```

javascript快速入门13--BOM——浏览器对象模型(Browser Object Model)

什么是BOM？

- BOM是Browser Object Model的缩写，简称浏览器对象模型
- BOM提供了独立于内容而与浏览器窗口进行交互的对象
- 由于BOM主要用于管理窗口与窗口之间的通讯，因此其核心对象是window
- BOM由一系列相关的对象构成，并且每个对象都提供了很多方法与属性
- BOM缺乏标准，JavaScript语法的标准化组织是ECMA，DOM的标准化组织是W3C（WHATWG,WebHypertextApplicationTechnologyWorkingGroup——网页超文本应用程序技术工作组目前正在努力促进BOM的标准化）
- BOM最初是Netscape浏览器标准的一部分

基本的BOM体系结构图



能利用BOM做什么？

BOM提供了一些访问窗口对象的一些方法，我们可以用它来移动窗口位置，改变窗口大小，打开新窗口和关闭窗口，弹出对话框，进行导航以及获取客户的一些信息如：浏览器品牌版本，屏幕分辨率。但BOM最强大的功能是它提供了一个访问HTML页面的一入口——document对象，以使得我们可以通过这个入口来使用DOM的强大功能！！！

window对象是BOM的顶层(核心)对象，所有对象都是通过它延伸出来的，也可以称为window的子对象。由于window是顶层对象，因此调用它的子对象时可以不显示的指明window对象，例如下面两行代码是一样的：

```
document.write("BOM");
window.document.write("BOM");
```

window -- window对象是BOM中所有对象的核心。window对象表示整个浏览器窗口，但不必表示其中包含的内容。此外，window还可用于移动或调整它表示的浏览器的大小，或者对它产生其他影响。

JavaScript中的任何一个全局函数或变量都是**window**的属性

window子对象

- document 对象
- frames 对象
- history 对象
- location 对象
- navigator 对象
- screen 对象

window对象关系属性

- parent：如果当前窗口为frame，指向包含该frame的窗口的frame （frame）
- self ：指向当前的window对象，与window同意。 （window对象）
- top ：如果当前窗口为frame，指向包含该frame的top-level的window对象
- window ：指向当前的window对象，与self同意。
- opener ：当窗口是用javascript打开时，指向打开它的那人窗口（开启者）

window对象定位属性

- IE提供了window.screenLeft和window.screenTop对象来判断窗口的位置，但未提供任何判断窗口大小的方法。用document.body.offsetWidth和document.body. offsetHeight属性可以获取视口的大小（显示HTML页的区域），但它们不是标准属性。
- Mozilla提供window.screenX和window.screenY属性判断窗口的位置。它还提供了window.innerWidth和window.innerHeight属性来判断视口的大小，window.outerWidth和window.outerHeight属性判断浏览器窗口自身的大小。

window对象的方法

窗体控制

moveBy(x,y)——从当前位置水平移动窗体x个像素，垂直移动窗体y个像素，x为负数，将向左移动窗体，y为负数，将向上移动窗体

moveTo(x,y)——移动窗体左上角到相对于屏幕左上角的(x,y)点，当使用负数做为参数时会吧窗体移出屏幕的可视区域

resizeBy(w,h)——相对窗体当前的大小，宽度调整w个像素，高度调整h个像素。如果参数为负值，将缩小窗体，反之扩大窗体

resizeTo(w,h)——把窗体宽度调整为w个像素，高度调整为h个像素

窗体滚动轴控制

scrollTo(x,y)——在窗体中如果有滚动条，将横向滚动条移动到相对于窗体宽度为x个像素的位置，将纵向滚动条移动到相对于窗体高度为y个像素的位置

scrollBy(x,y)—— 如果有滚动条，将横向滚动条移动到相对于当前横向滚动条的x个像素的位置(就是向左移动x像素)，将纵向滚动条移动到相对于当前纵向滚动条高度为y个像素的位置(就是向下移动y像素)

窗体焦点控制

focus()—— 使窗体或控件获取焦点

blur()——与focus函数相反，使窗体或控件失去焦点

新建窗体

open()——打开(弹出)一个新的窗体

close()——关闭窗体

opener属性——新建窗体中对父窗体的引用，中文"开启者"的意思

window.open方法语法

```
window.open(url, name, features, replace);
```

open方法参数说明

- url -- 要载入窗体的URL
- name -- 新建窗体的名称(目标,将在a 标签的target属性中用到，当与已有窗体名称相同时将覆盖窗体内容).open函数默认的打开窗体的方式为target的_blank弹出方式，因此页面都将以弹出的方式打开
- features -- 代表窗体特性的字符串，字符串中每个特性使用逗号分隔
- replace -- 一个布尔值，说明新载入的页面是否替换当前载入的页面，此参数通常不用指定

open函数features参数说明,如果不使用第三个参数，将打开一个新的普通窗口

参数名称	类型	说明
height	Number	设置窗体的高度，不能小于100
left	Number	说明创建窗体的左坐标，不能为负值
location	Boolean	窗体是否显示地址栏，默认值为no
resizable	Boolean	窗体是否允许通过拖动边线调整大小，默认值为no
scrollbars	Boolean	窗体中内部超出窗口可视范围时是否允许拖动，默认值为no
toolbar	Boolean	窗体是否显示工具栏，默认值为no
top	Number	说明创建窗体的上坐标，不能为负值
status	Boolean	窗体是否显示状态栏，默认值为no
width	Number	创建窗体的宽度，不能小于100

特性字符串中的每个特性使用逗号分隔，每个特性之间不允许有空格

open方法返回值为一个新窗体的window对象的引用

对话框

alert(str)—— 弹出消息对话框（对话框中有一个“确定”按钮）

confirm(str)—— 弹出消息对话框（对话框中包含一个“确定”按钮与“取消”按钮）

prompt(str,defaultValue)——弹出消息对话框（对话框中包含一个“确定”按钮、“取消”按钮与一个文本输入框），由于各个浏览器实现的不同，若没有第二个参数（文本框中的默认值）时也最好提供一个空字符串

状态栏

window.defaultStatus 属性——改变浏览器状态栏的默认显示(当状态栏没有其它显示时)，浏览器底部的区域称为状态栏，用于向用户显示信息

window.status 属性——临时改变浏览器状态栏的显示

时间等待与间隔函数

setTimeout()—— 暂停指定的毫秒数后执行指定的代码

clearTimeout()——取消指定的setTimeout函数将要执行的代码

setInterval()——间隔指定的毫秒数不停地执行指定的代码

clearInterval()——取消指定的setInterval函数将要执行的代码

setTimeout与setInterval方法有两个参数，第一个参数可以为字符串形式的代码，也可以是函数引用，第二个参数为间隔毫秒数,它们的返回是一个可用于对应clear方法的数字ID

```
var tid = setTimeout("alert('1')",1000);
   alert(tid);
   clearTimeout(tid);
```

History对象,在浏览器历史记录中导航

History 对象的属性:length 返回浏览器历史列表中的 URL 数量

History 对象的方法

- back() 加载 history 列表中的前一个 URL
- forward() 加载 history 列表中的下一个 URL
- go(num) 加载 history 列表中的某个具体页面

Location 对象

Location 对象的属性

- hash 设置或返回从井号 (#) 开始的 URL（锚）
- host 设置或返回主机名和当前 URL 的端口号
- hostname 设置或返回当前 URL 的主机名
- href 设置或返回完整的 URL
- pathname 设置或返回当前 URL 的路径部分
- port 设置或返回当前 URL 的端口号
- protocol 设置或返回当前 URL 的协议
- search 设置或返回从问号 (?) 开始的 URL（查询部分）

Location 对象的方法

- assign() 加载新的文档,这与直接将一个URL赋值给Location对象的href属性效果是一样的
- reload() 重新加载当前文档，如果该方法没有规定参数，或者参数是 false，它就会用 HTTP 头 If-Modified-Since 来检测服务器上的文档是否已改变。如果文档已改变，reload() 会再次下载该文档。如果文档未改变，则该方法将从缓存中装载文档。这与用户单击浏览器的刷新按钮的效果是完全一样的。如果把该方法的参数设置为 true，那么无论文档的最后修改日期是什么，它都会绕过缓存，从服务器上重新下载该文档。这与用户在单击浏览器的刷新按钮时按住 Shift 健的效果是完全一样。
- replace() 用新的文档替换当前文档，replace() 方法不会在 History 对象中生成一个新的纪录。当使用该方法时，新的 URL 将覆盖 History 对象中的当前纪录。

Navigator对象

Navigator 对象的属性

- appCodeName 返回浏览器的代码名

- appName 返回浏览器的名称
- appVersion 返回浏览器的平台和版本信息
- browserLanguage 返回当前浏览器的语言
- cookieEnabled 返回指明浏览器中是否启用 cookie 的布尔值
- cpuClass 返回浏览器系统的 CPU 等级
- onLine 返回指明系统是否处于脱机模式的布尔值
- platform 返回运行浏览器的操作系统平台
- systemLanguage 返回 OS 使用的默认语言
- userAgent 返回由客户机发送服务器的 user-agent 头部的值
- userLanguage 返回 OS 的自然语言设置

框架与多窗口通信

子窗口与父窗口

只有自身和使用window.open方法打开的窗口和才能被JavaScript访问,window.open方法打开的窗口通过window.opener属性来访问父窗口。 而在opener窗口中，可以通过window.open方法的返回值来访问打开的窗口！

框架

window.frames集合：在框架集或包含iframe标签的页面中，frames集合包含了对有框架中窗口的引用

```
    alert(frames.length);//框架的数目
    alert(frames[0].document.body.innerHTML);//使用下标直接获取对框架中窗口的引用
    //不但可以使用下标，还可以使用frame标签的name属性
    alert(frames["frame1"].document.title);
```

在框架集中还可以使用ID来获取子窗口的引用

```
 var frame1 =document.getElementById("frame1");//这样只是获取了标签
    var frame1Win = frame1.contentWindow;//frame对象的contentWindow包含了窗口的引用
    //还可以直接获取框架中document的引用
    var frameDoc = frame1.contentDocument;
    alert(frameDoc);//但IE不支持contentDocument属性
```

子窗口访问父窗口——**window**对象的**parent**属性

子窗口访问顶层——**window**对象的**top**属性

浏览器检测

市场上的浏览器种类多的不计其数，它们的解释引擎各不相同，期待所有浏览器都一致的支持JavaScript,CSS,DOM,那要等到不知什么时候，然而开发者不能干等着那天。历史上已经有不少方法来解决浏览器兼容问题了，主要分为两种：1.userAgent字符串检测，2.对象检测；当然，也不能考虑所有的浏览器，我们需要按照客户需求来，如果可以确信浏览网站的用户都使用或大部分使用IE浏览器，那么你大可放心的使用IE专有的那些丰富的扩展，当然，一旦用户开始转向另一个浏览，那么痛苦的日子便开始了。下面是市场上的主流浏览器列表：

- Internet Explorer
- Mozilla Firefox
- Google Chrome
- Opera
- Safari

注意，浏览器总是不断更新，我们不但要为多种浏览器作兼容处理，还要对同一浏览器多个版本作兼容处理。比如IE浏览器，其6.0版本和7.0版本都很流行，因为微软IE随着操作系统绑定安装（之前也是同步发行，微软平均每两年推出一款个人桌面，同样IE也每两年更新一次；直到现在，由于火狐的流行，IE工作组才加快IE的更新），所以更新的较慢，6.0版和7.0版有很大差别。

市场上还存在一些其它浏览器，但由于它们都是使用的上面所列浏览器的核心，或与上面浏览器使用了相同的解释引擎，所以无需多作考虑。下面是主流的浏览器解释引擎列表：

1. Trident

   Trident （又称为MSHTML），是微软的窗口操作系统（Windows）搭载的网页浏览器—Internet Explorer的排版引擎的名称，它的第一个版本随着1997年10月Internet Explorer第四版释出，之后不断的加入新的技术并随着新版本的Internet Explorer释出。在未来最新的Internet Explorer第七版中，微软将对Trident排版引擎做了的重大的变动，除了加入新的技术之外，并增加对网页标准的支持。尽管这些变动已经在相当大的程度上落后了其它的排版引擎。使用该引擎的主要浏览器：IE，TheWorld，MiniIE，Maxthon，腾讯TT浏览器。事实上，这些浏览器是直接使用了IE核心，因为其userAgent字符串中返回的信息与IE是一模一样的！

2. Gecko

   壁虎，英文为"Gecko"。Gecko是由Mozilla基金会开发的布局引擎的名字。它原本叫作NGLayout。Gecko的作用是读取诸如HTML、CSS、XUL和JavaScript等的网页内容，并呈现到用户屏幕或打印出来。Gecko已经被许多应用程序所使用，包括若干浏览器，例如Firefox、Mozilla Suite、Camino,Seamonkey等等

3. Presto

Presto是一个由Opera Software开发的浏览器排版引擎，供Opera 7.0及以上使用。Presto取代了旧版Opera 4至6版本使用的Elektra排版引擎，包括加入动态功能，例如网页或其部分可随着DOM及Script语法的事件而重新排版。Presto在推出后不断有更新版本推出，使不少错误得以修正，以及阅读Javascript效能得以最佳化，并成为速度最快的引擎。

4. KHTML

是HTML网页排版引擎之一，由KDE所开发。KDE系统自KDE2版起，在档案及网页浏览器使用了KHTML引擎。该引擎以C++编程语言所写，并以LGPL授权，支援大多数网页浏览标准。由于微软的Internet Explorer的占有率相当高，不少以FrontPage制作的网页均包含只有IE才能读取的非标准语法，为了使KHTML引擎可呈现的网页达到最多，部分IE专属的语法也一并支援。目前使用KHTML的浏览器有Safari和Google Chrome。而KHTML也产生了许多衍生品，如：WebKit,WebCore引擎

利用userAgent检测

下面是各大浏览器使用弹窗显示的userAgent字符串

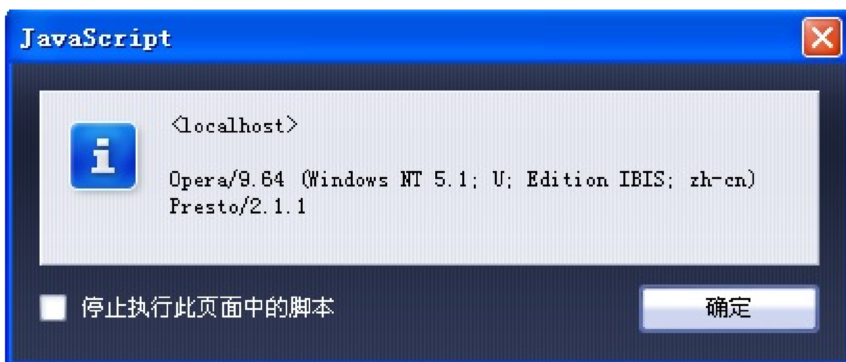
IE浏览器：Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727)



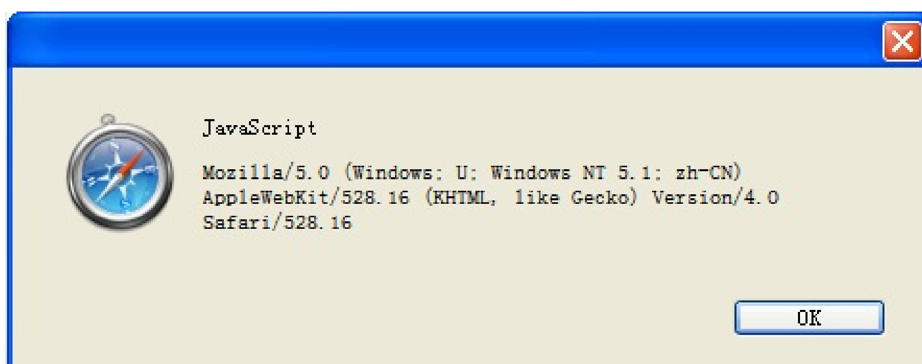
火狐浏览器：Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN; rv:1.9.0.4) Gecko/2008102920 Firefox/3.0.4



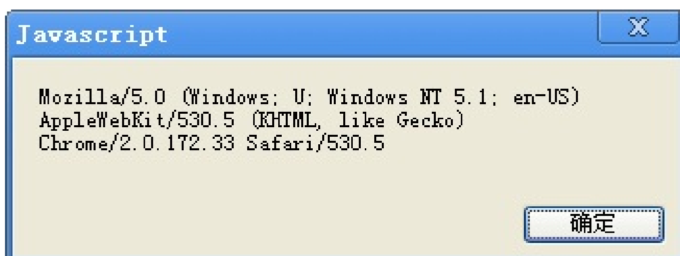
Opera浏览器：Opera/9.64 (Windows NT 5.1; U; Edition IBIS; zh-cn) Presto/2.1.1



Safari浏览器：Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN) AppleWebKit/528.16 (KHTML, like Gecko) Version/4.0 Safari/528.16



Google Chrome浏览器：Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/530.5 (KHTML, like Gecko) Chrome/2.0.172.33 Safari/530.5



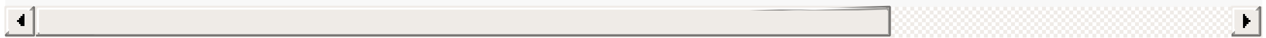
可以使用下面的代码进行浏览器检测

```
var Browser = {
       isIE:navigator.userAgent.indexOf("MSIE")!=-1,
       isFF:navigator.userAgent.indexOf("Firefox")!=-1,
       isOpera:navigator.userAgent.indexOf("Opera")!=-1,
       isSafari:navigator.userAgent.indexOf("Safari")!=-1 };
```

但这样做并不是万无一失的，一个特例便是Opera可以使用userAgent伪装自己。下面是伪装成IE的userAgent：Mozilla/5.0 (Windows NT 5.1; U; Edition IBIS; zh-cn; rv:1.8.1) Gecko/20061208 Firefox/2.0.0 Opera 9.64；在完全伪装的情况下，最后的“Opera 9.64”这个字符串也不会出现，但Opera也有特殊的识别自身的方法，它会自动声明一个opera全局变量！

不但如此，我们的检测还忽略了一点，就是那些使用相同引擎而品牌不同的浏览器，所以，直接检测浏览器是没有必要的，检测浏览器的解释引擎才是有必要的！

```
var Browser = {
        isIE:navigator.userAgent.indexOf("MSIE")>-1 && !window.opera,
        isGecko:navigator.userAgent.indexOf("Gecko")>-1 && !window.opera && navigator.use
        isKHTML:navigator.userAgent.indexOf("KHTML")>-1,
        isOpera:navigator.userAgent.indexOf("Opera")>-1 };
```



javascript快速入门14--DOM基础

DOM(Document Object Model)——文档对象模型

什么是DOM？

Document Object Model (DOM)是HTML和XML文档的编程接口。它提供了上述文档的一种结构化表示，同时也定义了一种通过程序来改变文档结构，风格，以及内容的方式。DOM用一组结构化的节点以及对象来表示文档。本质上就是将网页和脚本语言以及编程语言连接起来。

一个网页是一个文档。这个文档可以被显示在浏览器窗口中，也可以以html源码的形式显示。这两中情况中，文档都是同一个。DOM提供了另一种方式来表示，存储，操作这个文档。DOM是网页的一种完全的面向对象的表示方法，可以通过脚本语言（比如说JavaScript）来改变。

W3C DOM标准形成了当今大多数浏览器的DOM基础。许多浏览器提供超出W3C标准的扩展，所以当用在可能被拥有不同DOM的各种浏览器使用的场合时 一定要多加注意

DOM标准主要要为：微软DOM与W3C DOM，一般IE实现的是微软DOM，而其它浏览器则不同程度的实际了W3C DOM

DOM发展史

- DOM Level Zero ,事实上从来不存在DOM 0版本，只是人们的戏称。只是在W3C DOM出现之前，不同浏览器（主要是IE与NN）实现的DOM相互排斥，1996年的浏览器大战所产生的DHTML就是所谓的DOM 0，它是脚本程序员的恶梦
- DOM Level 1 包括DOM Core和DOM HTML。前者提供了基于XML的文档结构图。后者添加了一些HTML专用的对象和方法，从而扩展了DOM Core.目前IE在内的大部分桌面浏览器都通过不同方式实现了DOM 1
- DOM Level 2 引入几个新模块：DOM视图，事件，样式，遍历和范围。IE只实现了一部分，火狐浏览器几乎全部实现，除IE之外的浏览器也实现了大部分
- DOM Level 3 引入了以统一的方式载入和保存文档的方法。DOM Core被扩展支持所有的XML1.0的特性。火狐浏览器之类实现了少部分

在开始阶段，JavaScript和DOM是紧密的联系在一起的，但是最终他们将发展为独立的实体。网页的内容存储在DOM中并且可以被JavaScript访问和处理，所以我们可以得到写下这个近似的等式：

API（网页或者XML页面）＝DOM ＋ JS（脚本语言）

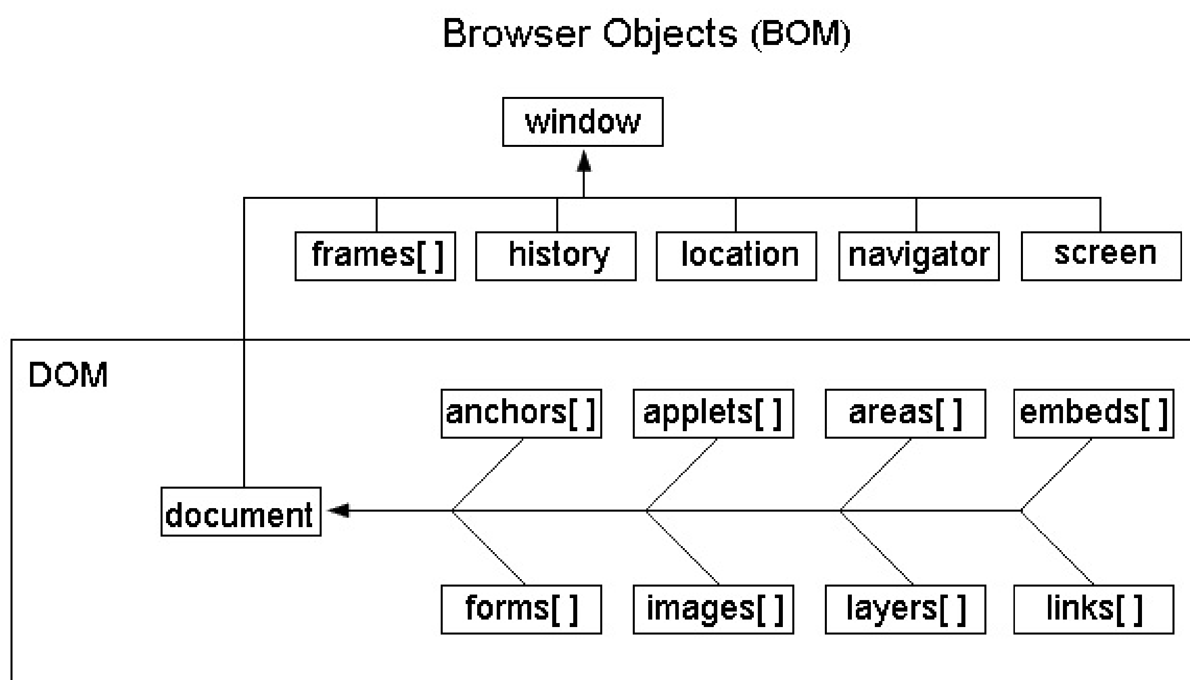
DOM被设计为独立于任何特定的编程语言，通过协调一致的API以确保这种文档的结构化表现形式是有效的。虽然DOM的实现可以建立在任何语言上，但是在这里我们专注于JavaScript上的DOM实现!

尽管DOM很大程度上受到浏览器中动态HTML发展的影响，但W3C还是将它最先应用于XML。

DOM与BOM的关系

DOM与BOM的关系？——BOM包含DOM

DOM与BOM结构视图

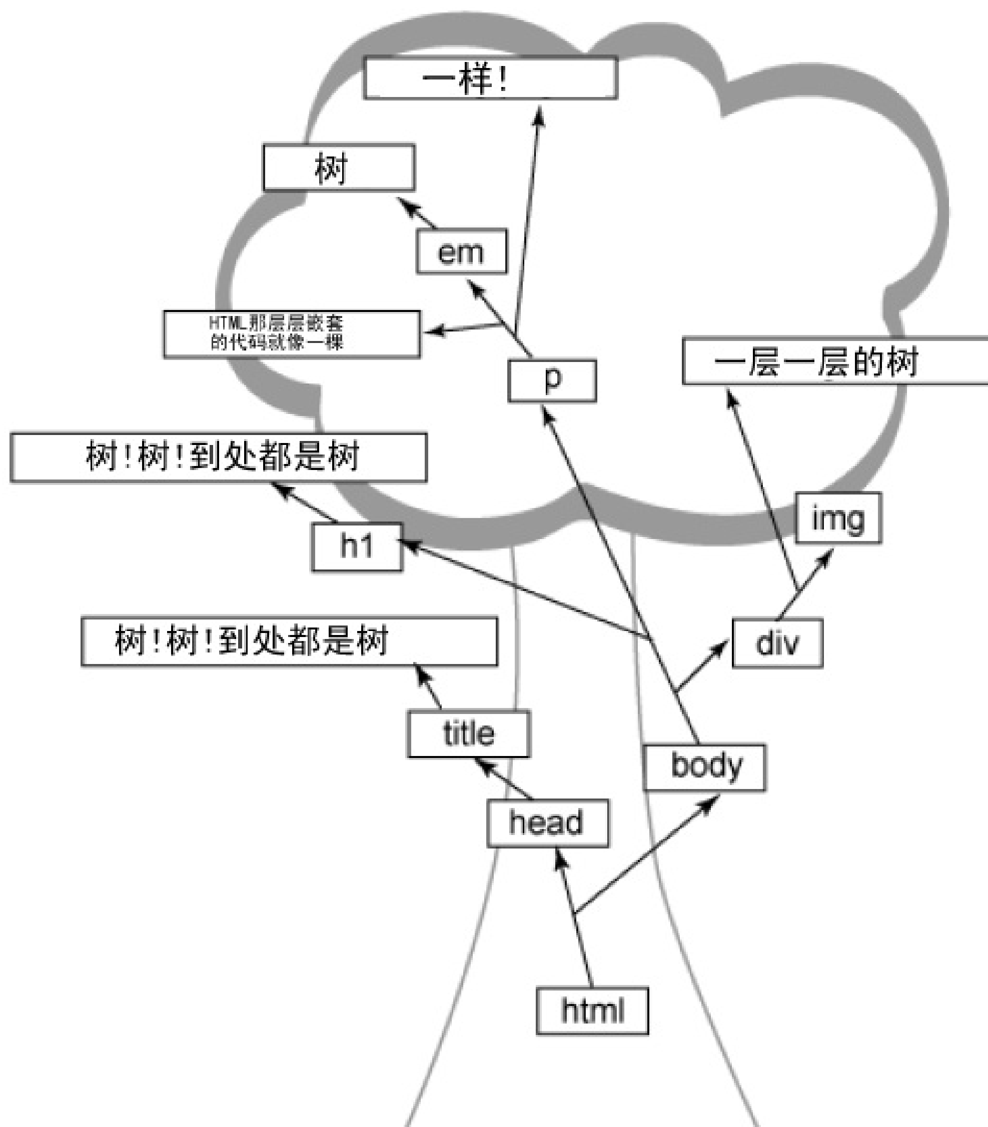


DOM Core

文档对象模型-DOM,就是使用树视图来形容HTML代码,看下面的一张HTML页面的源代码

```
<html>
       <head>
           <title>树!树!到处都是树!</title>
       </head>
       <body>
           <h1>树!树!到处都是树!</h1>
           <p>HTML那层层嵌套的代码就像一棵<em>树</em>一样!</p>
           <div>一层一层的树<img src="../images/stach_heap.gif" /> </div>
       </body>
   </html>
```

浏览器接受该页面并将之转换为树形结构



获取元素常用方法

document对象是BOM的一部分，同时也是HTML DOM的HTMLDocument对象的一种表现形式，反过来说，它也是XML DOM Document对象。JavaScript中的大部分处理DOM的过程都利用document对象，所以我们访问文档需要使用BOM提供的这个入口。

```
var d = document;
    alert(d);//这仅仅表明document这个对象是存在的
```

document对象有三个强大的方法，可以获取页面的任何元素

```
var p1 = document.getElementById("p1");//获取ID为p1的那个元素
   //在一个文档中ID必须是唯一的，getElementById方法只会返回一个元素
alert(p1.tagName); var allP = document.getElementsByTagName("p");//获取文档中所有p标签
   //因为页面中标签相同的元素很多,所以即使页面中只有一个p元素,getElementsByTagName也会返回一个集合
   for (var i=0;i < allP.length;i++) {
       alert(allP[i].innerHTML);//像数组一样访问其中的每个元素
} //getElementsByTagName还可以使用通配符*来获取所有的元素
   var allTags = document.getElementsByTagName("*");
   alert(allTags.length); //更强大的是，getElementsByTagName不但可以在document对象上调用，也可
   var p2 = document.getElementById("p2"); var p2ps = p2.getElementsByTagName("em");//将
   //还有一个通过name来获取元素的方法:getElementsByName
   var radios = document.getElementsByName("check");//获取所有name为check的元素
```

由于**name**可以重复，**getElementsByName**方法始终返回一个集合，不管页面中**name**是否是唯一的。 **IE 6.0**和**Opera 7.5**在这个方法的使用上还存在一些错误。首先，它们还会返回**id**等于给定名称的元素。第二，它们仅仅检查**<input/>**和**<img/>**元素。

获取和设置元素属性——getAttribute与setAttribute方法

```
var p1 = document.getElementById("p1");
   alert(p1.getAttribute("id"));
   p1.setAttribute("title","Value");
```

节点基础

文档根节点

```
var de = document.documentElement;
   alert(de.tagName);
```

由于**IE 5.5**中的**DOM**实现的错误，**document.documentElement**会返回**<body/>**元素。**IE 6.0**已经修复了这个错误。

获取head与body

```
//本来可以使用getElementsByTagName的
   var head = document.getElementsByTagName("head")[0]; var body = document.getElementsB
```

还可以使用节点,在使用节点前，先了解一些节点基础知识

常用节点类型

- 元素节点——文档中具有标签的节点
- 文本节点——标签中不是注释的文本块

常用的节点属性

- nodeType——节点类型，元素节点是1，文本节点是3

- nodeValue——节点值，元素节点为空，文本节点的nodeValue属性即为文本内容
- firstChild——该元素节点包含的第一个子节点
- lastChild——该元素节点包含的最后一个子节点
- nextSibling——该节点的后一个兄弟节点
- previousSibling——该节点的前一个兄弟节点
- childNodes——子节点列表,可以通过node.childNodes[index]（或node.childNodes.item(index)）来获取子节点
- nodeName——节点名称，对于元素节点，返回tagName,对于文本，则返回#text

考虑下面的HTML代码所表示的节点结构

```
<p> Some Text </p>
```

上面的HTML代码将会解析成两个节点:元素节点——p标签，文本节点——Some Text.也就是说，标签中包含的一些文本也是节点，那么空格之类的非打印字符会不会被当作文本节点呢？

不同浏览器在判断何为**Text**节点上存在一些差异。某些浏览器，如**Mozilla**，认为元素之间的空白**(**包括换行符**)**都是**Text**节点；而另一些浏览器，如**IE**，会全部忽略这些空白**!!**

```
var de = document.documentElement; var head = de.firstChild;//html下面第一个元素，可能是head
   var body = de.lastChild;//html下面最后一个元素，可能是body
```

答案并不确定，但是仍然有办法解决——使用节点类型检测,每个节点都有nodeType属性，指明它的节点类型。对于元素节点，它的值是1，而对于文本节点，它的值是3

```
var de = document.documentElement; var head = de.firstChild.nodeType==1?de.firstChild:de
   var de = document.documentElement; var head = de.childNodes[0].nodeType==1?de.childNo
```

HTML DOM

HTML DOM用对象来表示HTML标签,考虑下面的代码——

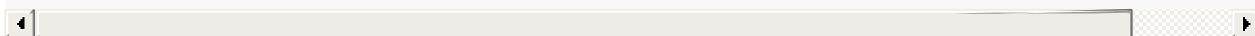
```
<img src="../images/stack_heap.jpg" alt="内存堆栈" onclick="alert('Hello!')"  /> //对于上面
   {
       src:"../images/stack_heap.jpg",
       alt:"内存堆栈",
       onclick:"alert('Hello!')",
       tagName:"IMG"
   };
   //其实不止这些属性
```

一般所说的DOM是指XML DOM，而W3C也为HTML页面提供了更快捷的DOM——HTML DOM!使用HTML DOM，能使访问HTML标签的属性就像访问JavaScript创建的对象的属性一样简单.

```
var imgObj = document.getElementById("imgObj");
   alert(imgObj.src);//获取src属性如此简单
```

使用HTML DOM也使得访问页面一些元素变得更加简单

```
var bodyTag = document.documentElement.lastChild;//DOM标准方式
   bodyTag = document.body;//HTML DOM方式
   var titleTag = document.getElementsByTagName("title")[0].firstChild.nodeValue;//DOM标
   titleTag = document.title;//HTML DOM方式
   //HTML DOM不仅仅可以用来获取内容，也可以设置
   document.title ="Change The Title!!!";
```



javascript快速入门15--节点

节点类型

DOM定义了Node的接口以及许多种节点类型来表示节点的多个方面!

- Document——最顶层的节点，所有的其他节点都是附属于它的。
- DocumentType——DTD引用（使用<!DOCTYPE >语法）的对象表现形式，例如<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" >。它不能包含子节点。
- DocumentFragment——可以像Document一样来保存其他节点。
- Element——表示起始标签和结束标签之间的内容，例如<tag ></tag >或者<tag / >。这是唯一可以同时包含特性和子节点的节点类型。
- Attr——代表一对特性名和特性值。这个节点类型不能包含子节点。
- Text——代表XML文档中的在起始标签和结束标签之间，或者CData Section内包含的普通文本。这个节点类型不能包含子节点。
- CDataSection——<![CDATA[ ]]>的对象表现形式。这个节点类型仅能包含文本节点Text作为子节点。
- Entity——表示在DTD中的一个实体定义，例如<!ENTITY foo "foo">。这个节点类型不能包含子节点。
- EntityReference——代表一个实体引用，例如"。这个节点类型不能包含子节点。
- ProcessingInstruction——代表一个PI(处理指令)。这个节点类型不能包含子节点。
- Comment——代表注释。这个节点类型不能包含子节点。
- Notation——代表在DTD中定义的记号。这个很少用到，所以我们不会讨论。

还定义了对应不同节点类型的12个常量.

- Node.ELEMENT_NODE (1)
- Node.ATTRIBUTE_NODE (2)
- Node.TEXT_NODE (3)
- Node.CDATA_SECTION_NODE (4)
- Node.ENTITY_REFERENCE_NODE (5)
- Node.ENTITY_NODE (6)
- Node.PROCESSING_INSTRUCTION_NODE (7)
- Node.COMMENT_NODE (8)
- Node.DOCUMENT_NODE (9)
- Node.DOCUMENT_TYPE_NODE (10)
- Node.DOCUMENT_FRAGMENT_NODE (11)
- Node.NOTATION_NODE (12)

节点类型常量都是**Node**对象的属性，但是**IE**不支持**Node**对象,但是仍可以使用数值

Node接口也定义了一些所有节点类型都包含的特性和方法。我们在下面的表格中列出了这些特性和方法：

特性/方法	类型/返回类型	说 明
nodeName	String	节点的名字；根据节点的类型而定义,元素节点返回tagName，文本节点返回#text,属性节点返回属性名
nodeValue	String	节点的值；根据节点的类型而定义.元素节点此属性为空，文本节点些属性即为节点中的字符串，属性节点返回属性值
nodeType	Number	节点的类型常量值之一
ownerDocument	Document	指向这个节点所属的文档
firstChild	Node	指向在childNodes列表中的第一个节点
lastChild	Node	指向在childNodes列表中的最后一个节点
childNodes	NodeList	所有子节点的列表
previousSibling	Node	指向前一个兄弟节点；如果这个节点就是第一个兄弟节点，那么该值为null
nextSibling	Node	指向后一个兄弟节点；如果这个节点就是最后一个兄弟节点，那么该值为null
hasChildNodes()	Boolean	当childNodes包含一个或多个节点时，返回眞
attributes	NamedNodeMap	包含了代表一个元素的特性的Attr对象；仅用于Element节点
appendChild(*node*)	Node	将*node*添加到childNodes的末尾
removeChild(*node*)	Node	从childNodes中删除*node*
replaceChild (*newnode*,*oldnode*)	Node	将childNodes中的*oldnode*替换成*newnode*
insertBefore (*newnode*,*refnode*)	Node	在childNodes中的*refnode*之前插入*newnode*

除节点外，DOM还定义了一些助手对象，它们可以和节点一起使用，但不是DOM文档必有的部分。

- NodeList——节点数组，按照数值进行索引；用来表示一个元素的子节点。比如childNodes。NodeList有个length属性表示节点数量
- NamedNodeMap——同时用数值和名字进行索引的节点表；用于表示元素特性。比如元素的attributes。NamedNodeMap对象也有一个length属性来指示它所包含的节点的数量。

这些助手对象为处理DOM文档提供附加的访问和遍历方法。

属性节点

正如前面提到的，即便Node接口已具有attributes方法，且已被所有类型的节点继承，然而，只有Element节点才能有特性。Element节点的attributes属性其实是NamedNodeMap，它提供一些用于访问和处理其内容的方法：

- getNamedItem(name)——返回nodeName属性值等于name的节点；
- removeNamedItem(name)——删除nodeName属性值等于name的节点；
- setNamedItem(node)——将node添加到列表中，按其nodeName属性进行索引；
- item(pos)——像NodeList一样，返回在位置pos的节点；

请记住这些方法都是返回一个**Attr**节点，而非特性值。

当NamedNodeMap用于表示特性时，其中每个节点都是Attr节点，它的nodeName属性被设置为特性名称，而nodeValue属性被设置为特性的值。示例：

```
<p id="p1" style="background-color:red;" title="P!!!">Some Text!</p> var p1 = document.g
   //访问ID属性
   alert(p1.attributes.getNamedItem("id")).nodeValue;
   //也可以用数值来访问ID属性
   alert(p1.attributes.item(0)).nodeValue;
   //也可以改变它的值
   p1.attributes.getNamedItem("id").nodeValue = "newP1";
```

Attr节点也有一个完全等同于（同时也完全同步于）nodeValue属性的value属性，并且有name属性和nodeName属性保持同步。我们可以随意使用这些属性来修改或变更特性。但这些方法都比较复杂，所以DOM又定义了三个元素方法来帮助访问特性：

- getAttribute(name)——等于attributes.getNamedItem(name).value
- setAttribute(name, newValue)——等于attribute.getNamedItem(name).value = newValue
- removeAttribute(name)——等于attributes.removeNamedItem(name)

NodeList

事实上我们早接触过NodeList了

```
var allTags = document.getElementsByTagName("*");
   alert(allTags.item(1).tagName);//访问了第二个元素
   alert(allTags[0]);//在JavaScript可以这样访问第一个元素，但这只能是JavaScript里正常运行
```

getElementsByTagName与getElementsByName都返回NodeList，可以使用item(index)方法访问其中的内容，在JavaScript中还可使用数组形式的下标访问！

创建和操纵节点

迄今为止，已经学过了如何访问文档中的不同节点，不过这仅仅是使用DOM所能实现的功能中的很小一部分。还能添加、删除、替换（或者其他操作）DOM文档中的节点。正是这些功能使得DOM具有真正意义上的动态性。

创建新节点

DOM Document（文档）中有一些方法用于创建不同类型的节点，即便在所有的浏览器中的浏览器document对象并不需要全部支持所有的方法。下面的表格列出了包含在DOM Level 1中的方法，并列出不同的浏览器是否支持项。

方法	描述	IE	MOZ	OP	SAF
createAttribute (*name*)	用给定名称*name*创建特性节点	×	×	×	—
createCDATASection (*text*)	用包含文本*text*的文本子节点创建一个CDATA Section	—	×	—	—
createComment(*text*)	创建包含文本*text*的注释节点	×	×	×	×
createDocumentFragment()	创建文档碎片节点	×	×	×	×
createElement (*tagname*)	创建标签名为*tagname*的元素	×	×	×	×
createEntity Reference(*name*)	创建给定名称的实体引用节点	—	×	—	—
createProcessing Instruction(*target*, *data*)	创建包含给定*target*和*data*的PI节点	—	×	—	—
createTextNode(*text*)	创建包含文本*text*的文本节点	×	×	×	×

注：IE = Windows的IE 6；MOZ = 任意平台的Mozilla 1.5；OP=任意平台的Opera 7.5；SAF=MacOS的Safari 1.2

最常用到的几个方法是：createDocumentFragment()、createElement()和createText-Node()；其他的一些方法要么就是没什么用（createComment()），要么就是浏览器的支持不够，目前还不太能用。

动态创建一个段落示例

```
var p = document.createElement("p");//创建一个元素节点，传入标签名
   var txt = document.createTextNode("创建文本节点,传参数即是文本内容");
   p.appendChild(txt);//将txt所引用的文本节点插入p到p的最后面(在这里p是空的)
   //直到现在，页面不会出现任何内容，必须将创建的节点插入到文档中
   document.body.appendChild(p);//p将出现在最后
```

移动，删除节点方法及注意事项——appendChild,removeChild,replaceChild,insertBefore

```
var p1 = document.getElementById("p1");
   document.body.appendChild(p1);//p1将会被作为body的最后一个子节点，然而页面上仍只有一个p
   p1.parentNode.removeChild(p1);//removeChild必须是要删除的节点的父节点调用
   //p1将会从页面上消失，然而它并没有完全消失，我们还可以再将其插入文档
   document.body.appendChild(p1);//因为变量p1包含了节点的引用
   var p2 = document.getElementById("p2");
   p2.parentNode.replaceChild(p1,p2);//p2将会被替换成p1，p2将消失
   //而p1将从原来的位置移到p2的位置
```

克隆节点——cloneNode

基于上面的原因，DOM为我们提供了一个克隆节点的方法用于生成一个节点的副本

```
var p1 = document.getElementById("p1"); var p2 = p1.cloneNode();
   document.body.appendChild(p2);//页面上将会多出一个段落,不过段落中什么都没有
   p2 = p1.cloneNode(true);//使用参数true表示克隆节点时包含子节点
   document.body.appendChild(p2);
```

javascript快速入门15--表单

大多数Web页面与用户之间的交互都发生在表单中，表单中有许多交互式HTML元素如：文本域，按钮，复选框，下拉列表等。从文档对象层次图中可以看到，表单是包含在文档中的，所以要访问表单，仍然需要通过document对象来访问

Form对象

表单就是指的form标签及其里面的内容，要获取一个表单对象，可以给某个form标签加个id属性，然后使用document.getElementById方法获得。也可以使用document.getElementsByTagName("form")来获取所有表单的集合，然后通过下标来访问。还可以给form标签加个name属性，然后可以使用document.getElementsByTagName来访问，注意，同样要使用下标来访问

事实上，0级DOM（0级DOM并不是任何DOM规范，事实上它是BOM的内容，但浏览器都实现的比较好）为我们提供了更简单的访问From对象的方法——使用document.formName

```
<form name="formName"></form> var fm = document.formName;//可以这样来直接引用该表单对象
   //与document.getElementsByTagName("form")相对应有document.forms集合
   var fm = document.forms[0];//获取第一个Form对象
```

访问表单元素

Form对象有个elements属性，包是一个含了form标签里面的所有表单控件（input,select等标签，但不包含如div之类的标签）的伪数组

```
var fm = document.forms[0];
   alert(fm.elements.length);//length属性报告了元素的个数
```

在之前，访问input这类标签和访问其它标签没什么区别，可以使用ID，也可以使用className,但当它们在表单中时，可以使用它们的nam有来访问

```
<form name="formName">
       <input name="textInput" type="text" value="文本框" />
   </form> alert(document.formName.textInput.value);
```

Form对象相关事件及方法

当表单提交时会发生submit事件，我们可以设置事件监听，当用户提交表单时检查表单内容。同时，如果用户输入有误，要阻止表单提交，可以在事件处理函数里return false就行了，当正确时可以调用表单的submit方法提交表单，使用表单的submit方法时不会执行submit

事件处理函数

```
    document.formName.onsubmit = function () { //检查表单
        if (result) { this.submit();
        } else { return false;
        }
    };
```

当表单被重置时会发生reset事件，但这个事件意义不大，因为reset按钮本身意义就不大。同时也有一个reset方法

```
    document.formName.onreset = function () { if (confirm("您真的要重置表单吗？")) { this.res
        } else { return false;
        }
    };
```

表单元素

单选按钮与复选框

单选按钮与复选框有个共同的属性——checked，指明该按钮是否被选中。而不同的是，往往多个单选按钮使用同一个name来分到相同的组，且只能有一个被选中，那么，使用这个name访问它时，由于多个按钮使用同一个名字，它会返回一个数组，而当只有一个时（事实不存在单选按钮只有一个单独存在的情况），它又会返回这个元素

```
 var radios = document.formName.radios;//页面中多个单选按钮name为radios
    alert(radios.length);//返回一个元素列表
    var one = document.formName.one;//只有一个
    alert(one.checked);//只返回这个元素
```

与checked类似的，它们还有个defaultChecked属性，返回是否是在HTML指定默认选中的

单行文本框与多行文本框

单行文本框即type属性设为text的input标签，多行文本框即textarea，它们除了与其它input标签一样具有的value属性处，还具有defaultValue属性表示文本框中的默认文本，即在HTML里所指定的value值的

文本框还有一个方法：select,可以使文本框中的文字呈选中状态。

Select对象

表单元素中最复杂的就算是select对象了。select是一复合对象，它包含option标签，也有可能包含optgroup标签。虽然select可以多选，但我们这里只讨论单选的，多选的类似！

```
//首先，获取select对象也是通过name（当然ID仍然有效，但name更快捷）
   var select = document.formName.mySel; //要获取用户选中了第几项，可以使用它的selectedIndex
   alert(select.selectedIndex);//这个索引号是从0开始的
   //select对象有个options伪数组类型的属性，包含了所有的option，可以使用下标来访问
   alert(select.options[select.selectedIndex].value);//输出选中项的
   alert(select.options[select.selectedIndex].text);//text属性是option包含的文本
   //本来需要知道select里面有多少个option，可以通过options.length
   //但HTML DOM为select对象也提供了length属性
   alert(select.length);//输出和options.length一样
```

获取选中项的值

```
var mySel = document.formName.mySel;
   alert(mySel.options[mySel.selectedIndex].value); //更快捷的方法
   alert(mySel.value);
```

而向select对象中添加option，本可以使用document.createElement及appendChild等方法的，但HTML DOM为我们提供了更方便的方法了

```
var opt = new Option("新增选项文本","选项值");//document.createElement("option");
   var select = document.formName.mySel;
   select.add(opt,select.options[0]);//将opt添加到第一个option的后面
   select.add(opt);//在IE下，没有第二个参数时，会将opt添加到最后
   //上面的代码在FF下会出错，必须使用下面的方法
   select.add(opt,null);//但这在IE会出错
   //下面的方法可以两全了,但长了一点
   select.add(opt,select.options[select.length-1]); //删除option的方法remove
   select.remove(1);//remove接收参数为要移除
```

options数组最特殊的一个地方在于它和真正的数组十分相似，可以设置它的length来减少元素，也可以直接将元素赋给某项来修改

```
var opts = document.formName.mySel.options;
   opts[0]=new Option("Text","Value");
   opts.length=2;//将移去第三个之后的option
   opts[3]=new Options("ABC",123);//自动添加一个元素
```

Option对象也有defaultSelected属性返回在HTML里指定是否是默认选中项

表单元素特性事件

当表单控件失去焦点时会触发blur事件，当控件获得焦点时又会触发focus事件。与之对应，blur方法将使表单控件失去焦点，focus方法将使控件获得焦点，与Form对象的submit方法一样，使用JavaScript执行blur方法与focus方法并不会触发相关事件监听函数

select对象的change事件：当用户选中一个选项，或者取消了对一个选项的选定时，就会发生change事件。

textarea对象的change事件：当用户改变文本区域的值然后通过把键盘焦点移动到其他地方“确认”这些改变的时候，发生change事件。

select事件：当文本框中的文本被选中时发生

javascript快速入门16--表格

表格的层次结构

```
<table border="1">
    <caption>表格标题</caption>
    <thead>
        <tr>
            <th>表头1</th>
            <th>表头2</th>
            <th>表头3</th>
        </tr>
    </thead>
    <tfoot>
        <tr>
            <td colspan="3">脚注</td>
        </tr>
    </tfoot>
    <tbody>
        <tr>
            <td>数据</td>
            <td>数据</td>
            <td>数据</td>
        </tr>
    </tbody>
    <tbody>
        <tr>
            <td>数据</td>
            <td>数据</td>
            <td>数据</td>
        </tr>
    </tbody>
</table>
```

上面是一个创建表格所用到的所有标签，但一些标签是可写可不写的，事实上一般的表格只需写上tr与td标签就行了，而标题caption,表头thead表尾tbody等则是一些语义性元素

表格对象的一些属性

```
 var table= document.getElementById("myTable"); //获取表格标题caption标签
    var caption = table.getElementsByTagName("caption")[0]; //HTML DOM提供的更简单的方法
    caption= table.caption;//返回表格标题caption标签，如果没有则返回null
    if (caption) {
        alert(caption.firstChild.nodeValue);//输出标题文本
 } var thead =table.tHead;//获取表头
    var tfoot = table.tFoot;//获取表尾
```

由于caption,thead,tfoot这些标签一个表格中只能出现一次，所以HTML DOM提供了直接的属性来访问，而对于tr,td,th,tbody这些重复的标签，HTML DOM则给Table对象增加了一些集合来访问

```
//获取所有tr
   var rows = table.getElementsByTagName("tr");//但会获取嵌套表格中的tr
   //rows集合只会包含表格的行，而不包含表格下面嵌套表格的行
   rows = table.rows;//返回包含表格中所有行的一个数组
   alert(rows[0].innerHTML); var tBodies = table.tBodies;//返回包含表格中所有tbody的一个数组
   var cells =table.cells;//返回包含表格中所有单元格的一个数组
```

注意，对于**Table**对象的**cells**属性，它将返回所有**td,th**标签，而对于**tBodies**属性，即使**HTML** 源代码中没有**tbody**标签，也会默认有一个**tbody**

表格对象的一些方法

创建标题:createCaption() 方法用于在表格中获取或创建元素。返回一个 HTMLElement 对象，表示该表的元素。如果该表格已经有了标题，则返回它。如果该表没有元素，则创建一个新的空元素，把它插入表格，并返回它。

```
var caption = document.createElement("caption");
   caption.appendChild(document.createTextNode("新标题"));
   table.insertBefore(caption,table.firstChild); /* 上面方法有两大缺点：
   1.方法复杂
   2.如果已经存在caption标签，则会造成caption标签重复，导致后插入的无效 */ caption = table.create
   //因为该方法必须在对应的表格对象上调用
   caption.innerHTML = "新标题";
```

与createCaption相似的还有：

- createTFoot() 在表格中创建一个空的 tFoot 元素;返回一个 TableSection，表示该表的〈tfoot〉 元素。如果该表格已经有了脚注，则返回它。如果该表没有脚注，则创建一个新的空 〈tfoot〉 元素，把它插入表格，并返回它。
- createTHead() 在表格中创建一个空的 tHead 元素;返回一个 TableSection，表示该表的〈thead〉元素。如果该表格已经有了表头，则返回它。如果该表没有表头，则创建一个新的空〈thead〉元素，把它插入表格，并返回它。

既然有增加的方法，就有对应的删除的方法

- deleteCaption() 从表格删除 caption 元素以及其内容。 如果该表有 〈caption〉 元素，则从文档树种删除它。否则，什么也不做。
- deleteTFoot() 从表格删除 tFoot 元素及其内容。 如果该表有 〈tfoot〉 元素，则将它从文档树种删除，否则什么也不做。
- deleteTHead() 方法用于从表格删除thead 元素。如果该表有 〈thead〉元素，则将它从文档树种删除，否则什么也不做。

添加与删除行

- insertRow() 在表格中插入一个新行。 返回一个 TableRow，表示新插入的行。该方法创建一个新的 TableRow 对象，表示一个新的〈tr〉标记，并把它插入表中的指定位置。新

行将被插入 index 所在行之前。若 index 等于表中的行数，则新行将被附加到表的末尾。如果表是空的，则新行将被插入到一个新的〈tbody〉 段，该段自身会被插入表中。
- deleteRow() 从表格删除一行。参数 index 指定了要删除的行在表中的位置。行的编码顺序就是他们在文档源代码中出现的顺序。〈thead〉和〈tfoot〉 中的行与表中其它行一起编码。

行 (TableRow) 对象

行对象的一些属性：cells属性返回行中所有单元格的一个数组。rowIndex属性返回该行在表中的位置。sectionRowIndex属性返回在 tBody 、tHead 或 tFoot 中，行的位置。

```
var row = table.rows[0];
   alert(row.cells.length);//第一行中单元格的数目
   alert(row.rowIndex);//0
```

TableRow 对象的方法

- deleteCell() 删除行中的指定的单元格。参数 index 是要删除的表元在行中的位置。该方法将删除表行中指定位置的表元。
- insertCell() 在一行中的指定位置插入一个空的td元素。 返回一个 TableCell 对象，表示新创建并被插入的 td 元素。 该方法将创建一个新的 td 元素，把它插入行中指定的位置。新单元格将被插入当前位于 index 指定位置的表元之前。如果 index 等于行中的单元格数，则新单元格被附加在行的末尾。请注意，该方法只能插入 td 数据表元。若需要给行添加头表元，必须用 Document.createElement() 方法和 Node.insertBefore() 方法（或相关的方法）创建并插入一个 th 元素。

```
var row = table.rows[2]; var cell = row.insertCell(2);
   cell.innerHTML = "新插入的单元格"; //上面的代码与下面的等效(但不考虑空白文本节点)
   var cell = document.createElement("td");
   cell.innerHTML = "新插入的单元格";
   row.insertBefore(cell,row.childNodes[2]); //删除单元格
   row.deleteCell(2); //等效代码(同样不考虑空白文本节点)
   row.removeChild(row.childNodes[2]);
```

TableCell 单元格对象

与TableCell对象相关的有用的属性只有一个：cellIndex属性返回单元在格行中的下标

```
   alert(table.rows[2].cells[3].cellIndex);//3
```

javascript快速入门17--事件

事件(上)

JavaScript事件列表		
事件	解说	
一般事件	onclick	鼠标点击时触发此事件
ondblclick	鼠标双击时触发此事件	
onmousedown	按下鼠标时触发此事件	
onmouseup	鼠标按下后松开鼠标时触发此事件	
onmouseover	当鼠标移动到某对象范围的上方时触发此事件	
onmousemove	鼠标移动时触发此事件	
onmouseout	当鼠标离开某对象范围时触发此事件	
onkeypress	当键盘上的某个键被按下并且释放时触发此事件.	
onkeydown	当键盘上某个按键被按下时触发此事件	
onkeyup	当键盘上某个按键被按放开时触发此事件	
页面相关事件	onabort	图片在下载时被用户中断
onbeforeunload	当前页面的内容将要被改变时触发此事件	
onerror	出现错误时触发此事件	
onload	页面内容完成时触发此事件	
onmove	浏览器的窗口被移动时触发此事件	
onresize	当浏览器的窗口大小被改变时触发此事件	
onscroll	浏览器的滚动条位置发生变化时触发此事件	
onstop	浏览器的停止按钮被按下时触发此事件或者正在下载的文件被中断	
oncontextmenu	当弹出右键上下文菜单时发生	
onunload	当前页面将被改变时触发此事件	
		当前元素失去焦点时

		触发此事件
onchange	当前元素失去焦点并且元素的内容发生改变而触发此事件	
onfocus	当某个元素获得焦点时触发此事件	
onreset	当表单中RESET的属性被激发时触发此事件	
onsubmit	一个表单被递交时触发此事件	

了解上面的事件如此简单，那么事件还有什么可讲的呢？

问题一：每个事件只能注册一个函数

```
var oDiv = document.getElementById("oDiv");
   oDiv.onclick = fn1;
   oDiv.onclick =fn2; function fn1() {alert("我被覆盖了！")} function fn2() {alert("只有我
```

解决方案一：

```
   obj.onclick = function () {
       fn1();
       fn2();
       fn3();
   };
```

缺陷一：需要将所有函数一次添加进去，不能在运行时添加

缺陷二：在事件处理函数中this将指向window,而不是obj

解决方案二：

```
function addEvent(fn,evtype,obj) { //obj是要添加事件的HTML元素对象
       //evtype是事件名字，不包含on前缀，因为每个都有on，所以写个on是多余的
       //fn是事件处理函数
       var oldFn; if (obj["on"+evtype] instanceof Function) {
           oldFn = obj["on"+evtype];//当添加函数时，如果已注册过了，则将其保存起来
}
       obj["on"+evtype]=function () { if (oldFn) {
                   oldFn.call(this);
               }
               fn.call(this);//使用call方法，使事件处理函数中的this仍指向obj
};
   }
```

这样已经解决了问题，但如何删除事件呢？如果直接将对象的onevtype这类的属性赋值为null将会删除所有的事件处理函数！

解决方案二的修改版：先将事件存储起来，存储在对象的__EventHandles属性里面

```
    eventHandlesCounter=1;//计数器，将统计所有添加进去的函数的个数,0位预留作其它用
    function addEvent(fn,evtype,obj) { if (!fn.__EventID) {//__EventID是给函数加的一个标识，
            fn.__EventID=eventHandlesCounter++; //使用一个自动增长的计数器作为函数的标识以保证不
 } if (!obj.__EventHandles) {
            obj.__EventHandles=[];//当不存在，也就是第一次执行时，创建一个，并且是数组
 } if (!obj.__EventHandles[evtype]) {//将所有事件处理函数按事件类型分类存放
            obj.__EventHandles[evtype]=[];//当不存在时也创建一个数组
            if (obj["on"+evtype] instanceof Function) { //查看是否已经注册过其它函数
                //如果已经注册过，则将以前的事件处理函数添加到数组下标为0的预留的位置
                obj.__EventHandles[evtype][0]=obj["on"+evtype];
                obj["on"+evtype]=handleEvents;//使用handleEvents集中处理所有的函数
 }
        }
        obj.__EventHandles[evtype][fn.__EventID]=fn; //如果函数是第一次注册为事件处理函数，那么
        //如果函数已经注册过相同对象的相同事件了，那么将覆盖原来的而不会被添加两次
        function handleEvents() { var fns = obj.__EventHandles[evtype]; for (var i=0;i< f
                fns[i].call(this);
            }
        }
    }
```

使用上面的函数已经可以在一个对象添加多个事件处理函数，在函数内部this关键字也指向了相应的对象，并且这些函数都被作了标识，那么移除某个事件处理函数就是轻而易举的了！

```
 //使用传统方法：obj.onevtype = null;但这样会移除所有的事件处理函数
    function delEvent(fn,evtype,obj) { if (!obj.__EventHandles || !obj.__EventHandles[evt
        } if (obj.__EventHandles[evtype][fn.__EventID] == fn) { delete obj.__EventHandles
        }
    }
```

事件(下)

事件对象——Event

事件对象是用来记录一些事件发生时的相关信息的对象。事件对象只有事件发生时才会产生，并且只能是事件处理函数内部访问，在所有事件处理函数运行结束后，事件对象就被销毁！

访问事件对象：W3C DOM方法与IE专用方法

```
 //W3C DOM把事件对象作为事件处理函数的第一个参数传入进去
    document.onclick = function (evt) {//这样，事件对象只能在对应的事件处理函数内部可以访问到
 alert(evt);
    }; //IE将事件对象作为window对象的一个属性（相当于全局变量）
    //貌似全局对象，但是只有是事件发生时才能够访问
    alert(window.event);//null
    window.onload = function () {
        alert(window.event);
    };
```

事件对象的属性及方法

属性名	值类型	读/写	描述
button	Integer	R	对于特定的鼠标事件，表示按下的鼠标按钮，该属性仅可以在mouseup与mousedown事件中访问。W3C 规定：0表示按下了左键，1表示按下了中键，2表示按下了右键，相当于对于鼠标键从左到右进行的编号，而编号从0开始； 而IE有另外一套规定：0表示没有任何键按下，1表示左键，2表示右键，4表示中键，而其它按键的组合则只要将键码相加即可，如：同时按下左右键时button值为3
clientX	Integer	R	事件发生时，鼠标在客户端区域的X坐标，客户端区域是指页面可视区域
clientY	Integer	R	事件发生时，鼠标在客户端区域的Y坐标
screenX	Integer	R(IE)R/W(W3C)	相对于屏幕的鼠标X坐标
screenY	Integer	R(IE)R/W(W3C)	相对于屏幕的鼠标Y坐标
x(仅IE)	Integer	R	鼠标相对于引起事件的元素的父元素的X坐标
y(仅IE)	Integer	R	鼠标相对于引起事件的元素的父元素的Y坐标
offsetX(仅IE)layerX(仅W3C)	Integer	R	鼠标相对于引起事件的对象的X坐标
offsetY(仅IE)layerY(仅W3C)	Integer	R	鼠标相对于引起事件的对象的Y坐标
pageX(仅W3C)	Integer	R	鼠标相对于页面的X坐标
pageY(仅W3C)	Integer	R	鼠标相对于页面的Y坐标

属性名	值类型	读/写	描述
altKey	Boolean	R	true表示按下了ALT键；false表示没有按下
ctrlKey	Boolean	R	true表示按下了CTROL，false表示没有
shiftKey	Boolean	R	true表示按下了shift，false表示没有
keyCode	Integer	R/W(IE)R(W3C)	对于keypress事件，表示按下按钮的Unicode字符；对于keydown/keyup事件，表示按下按钮的数字代号
charCode(仅W3C)	Integer	R	在keypress事件中所按键的字符Unicode编码，如果不是字符键，则该属性为0，并且，当CapsLock打开与关闭时charCode的值也对应着大小写字母

其它

属性名	值类型	读/写	描述
srcElement(IE)target(W3C)	Element	R	引起事件的元素
fromElement(仅IE)	Element	R	某些鼠标事件中(mouseover与mouseout)，鼠标所离开的元素
toElement(仅IE)	Element	R	某些鼠标事件中(mouseover与mouseout)，鼠标所进入的元素
relatedTarget(仅W3C)	Element	R	某些鼠标事件中(mouseover与mouseout)，返回与事件的目标节点相关的节点。
repeat(仅IE)	Boolean	R	如果不断触发keydown事件，则为true,否则为false
returnValue(仅IE)	Boolean	R/W	将其设为false表示以取消事件的默认动作
preventDefault(仅W3C)	Function	R	执行方法以取消事件的默认动作
type	String	R	事件的名称，不带on前缀
cancelable(仅W3C)	Boolean	R	当为true表示事件的默认动作可以被取消（用preventDefault方法取消）
cancelBubble(仅IE)	Boolean	R/W	将其设置为true将取消事件冒泡
stopPropagation(仅W3C)	Function	R	执行方法取消事件冒泡
bubbles(仅W3C)	Boolean	R	返回true表示事件是冒泡类型
eventPhase(仅W3C)	Integer	R	返回事件传播的当前阶段。它的值是下面的三个常量之一，它们分别表示捕获阶段、在目标对象上时和起泡阶段：

| 常量 | 值 | | --- | --- | | Event.CAPTURING_PHASE(捕获阶段) | 1 | | Event.AT_TARGET(在目标对象上) | 2 | | Event.BUBBLING_PHASE(冒泡阶段) | 3 |

| | timeStamp (仅W3C) | Long | R | 返回一个时间戳。指示发生事件的日期和时间（从 epoch 开始的毫秒数）。epoch 是一个事件参考点。在这里，它是客户机启动的时间。并非所有系统都提供该信息，因此，timeStamp 属性并非对所有系统/事件都是可用的。 |

取得事件对象及取得事件目标对象

```
document.onclick =function (evt) {
    evt = evt || window.event;//在IE中evt会是undefined
    //而支持W3C DOM事件的浏览器中事件对象将会作为事件处理函数的第一个参数
    var targetElement = evt.target || evt.srcElement; //IE中事件对象没有target属性
};
```

阻止事件发生时浏览器的默认行为

```
    document.onclick = function (evt) {
        evt = evt || window.event; var target = evt.target || evt.srcElement; if (!target
        } if (target.tagName=="A" && target.href) { //使用传统的方法取消事件默认行为必须使用ret
            //但使用了return ,函数便终止了运行，可以使用事件对象来取消
            if (window.event) {//IE
                window.event.returnValue = false;
            } else {
                evt.preventDefault();
            }
            window.open(target.href,"newWindow"); //这样让所有的链接在新窗口打开
 }
    };
```

事件传播——冒泡与捕获

DOM事件标准定义了两种事件流，这两种事件流有着显著的不同并且可能对你的应用有着相当大的影响。这两种事件流分别是捕获和冒泡。和许多Web技术一样，在它们成为标准之前，Netscape和微软各自不同地实现了它们。Netscape选择实现了捕获事件流，微软则实现了冒泡事件流。幸运的是，W3C决定组合使用这两种方法,并且大多数新浏览器都遵循这两种事件流方式。

默认情况下，事件使用冒泡事件流，不使用捕获事件流。然而，在Firefox和Safari里，你可以显式的指定使用捕获事件流，方法是在注册事件时传入useCapture参数，将这个参数设为true。

冒泡事件流

当事件在某一DOM元素被触发时，例如用户在客户名字节点上点击鼠标，事件将跟随着该节点继承自的各个父节点冒泡穿过整个的DOM节点层次，直到它遇到依附有该事件类型处理器的节点，此时，该事件是onclick事件。在冒泡过程中的任何时候都可以终止事件的冒泡，在遵从W3C标准的浏览器里可以通过调用事件对象上的stopPropagation()方法，在Internet Explorer里可以通过设置事件对象的cancelBubble属性为true。如果不停止事件的传播，事件将一直通过DOM冒泡直至到达文档根。

捕获事件流

事件的处理将从DOM层次的根开始，而不是从触发事件的目标元素开始，事件被从目标元素的所有祖先元素依次往下传递。在这个过程中，事件会被从文档根到事件目标元素之间各个继承派生的元素所捕获，如果事件监听器在被注册时设置了useCapture属性为true,那么它们可以被分派给这期间的任何元素以对事件做出处理；否则，事件会被接着传递给派生元素路径上的下一元素，直至目标元素。事件到达目标元素后，它会接着通过DOM节点再进行冒泡。

现代事件绑定方法

针对如上节课所讨论的，使用传统事件绑定有许多缺陷，比如不能在一个对象的相同事件上注册多个事件处理函数。而浏览器和W3C也并非没有考虑到这一点，因此在现代浏览器中，它们有自己的方法绑定事件。

W3C DOM

- obj.addEventListener(evtype,fn,useCapture)——W3C提供的添加事件处理函数的方法。obj是要添加事件的对象，evtype是事件类型，不带on前缀，fn是事件处理函数，如果useCapture是true，则事件处理函数在捕获阶段被执行，否则在冒泡阶段执行
- obj.removeEventListener(evtype,fn,useCapture)——W3C提供的删除事件处理函数的方法

微软IE方法

- obj.attachEvent(evtype,fn)——IE提供的添加事件处理函数的方法。obj是要添加事件的对象，evtype是事件类型，带on前缀，fn是事件处理函数，IE不支持事件捕获
- obj.detachEvent(evtype,fn,)——IE提供的删除事件处理函数的方法，evtype包含on前缀

整合两者的方法

```
function addEvent(obj,evtype,fn,useCapture) { if (obj.addEventListener) {
            obj.addEventListener(evtype,fn,useCapture);
        } else {
            obj.attachEvent("on"+evtype,fn);//IE不支持事件捕获
        } else {
            obj["on"+evtype]=fn;//事实上这种情况不会存在
}
    } function delEvent(obj,evtype,fn,useCapture) { if (obj.removeEventListener) {
            obj.removeEventListener(evtype,fn,useCapture);
        } else {
            obj.detachEvent("on"+evtype,fn);
        } else {
            obj["on"+evtype]=null;
        }
    }
```

其它兼容性问题：IE不支持事件捕获？很抱歉，这个没有办法解决！但IE的attach方法有个问题，就是使用attachEvent时在事件处理函数内部，this指向了window，而不是obj！当然，这个是有解决方案的！

```
function addEvent(obj,evtype,fn,useCapture) { if (obj.addEventListener) {
            obj.addEventListener(evtype,fn,useCapture);
        } else {
            obj.attachEvent("on"+evtype,function () {
                fn.call(obj);
            });
        } else {
            obj["on"+evtype]=fn;//事实上这种情况不会存在
}
    }
```

但IE的attachEvent方法有另外一个问题，同一个函数可以被注册到同一个对象同一个事件上多次，解决方法：抛弃IE的attachEvent方法吧！IE下的attachEvent方法不支持捕获，和传统事件注册没多大区别(除了能绑定多个事件处理函数)，并且IE的attachEvent方法存在内存泄漏问题！

addEvent,delEvent现代版

```
function addEvent(obj,evtype,fn,useCapture) { if (obj.addEventListener) {//优先考虑W3C事件
            obj.addEventListener(evtype,fn,!!useCapture);
        } else {//当不支持addEventListener时(IE),由于IE同时也不支持捕获,所以不如使用传统事件绑定
            if (!fn.__EventID) {fn.__EventID = addEvent.__EventHandlesCounter++;} //为每个

            if (!obj.__EventHandles) {obj.__EventHandles={};} //__EventHandles属性用来保存所

            //按事件类型分类
            if (!obj.__EventHandles[evtype]) {//第一次注册某事件时
                obj.__EventHandles[evtype]=[]; if (obj["on"+evtype]) {//以前曾用传统方式注册
                    (obj.__EventHandles[evtype][0]=obj["on"+evtype]).__EventID=0;//添加到预
                    //并且给原来的事件处理函数增加一个ID
}
                obj["on"+evtype]=addEvent.execEventHandles; //当事件发生时，execEventHandle
}
        }
    }
    addEvent.__EventHandlesCounter=1;//计数器,0位预留它用
    addEvent.execEventHandles = function (evt) {//遍历所有的事件处理函数并执行
        if (!this.__EventHandles) {return true;}
        evt = evt || window.event; var fns = this.__EventHandles[evt.type]; for (var i=0;
                fns[i].call(this);
            }
        }
    }; function delEvent(obj,evtype,fn,useCapture) { if (obj.removeEventListener) {//先使
            obj.removeEventListener(evtype,fn,!!useCapture);
        } else { if (obj.__EventHandles) { var fns = obj.__EventHandles[evtype]; if (fns)
            }
        }
    }
```

标准化事件对象

IE的事件对象与W3C DOM的事件对象有许多不一样的地方，解决的最好的方法就是调整IE的事件对象，以使它尽可能的与标准相似！下表列出了IE事件对象中一些和W3C DOM名称或值不一样但含义相同的属性

W3C DOM	IE
button——按键编码为：0-左键，1-中键，2-右键	button——按键编码为：1-左键，2-右键，4-中键
charCode	没有对应属性，但可以用keyCode来代替
preventDefault	没有对应方法，但可以将event对象的returnValue设为false来模拟
target	srcElement
relatedTarget	fromElement与toElement
stopPropagation	没有对应方法，但可以通过将event对象的cancelBubble属性设为true来模拟

总结出fixEvent函数

```
function fixEvent(evt) { if (!evt.target) {
           evt.target = evt.srcElement;
           evt.preventDefault = fixEvent.preventDefault;
           evt.stopPropagation = fixEvent.stopPropagation; if (evt.type == "mouseover")
               evt.relatedTarget = evt.fromElement;
           } else if (evt.type =="mouseout") {
               evt.relatedTarget = evt.toElement;
           }
           evt.charCode =  (evt.type=="keypress")?evt.keyCode:0;
           evt.eventPhase = 2;//IE仅工作在冒泡阶段
           evt.timeStamp = (new Date()).getTime();//仅将其设为当前时间
} return evt;
   }
   fixEvent.preventDefault =function () { this.returnValue = false;//这里的this指向了某个事
};
   fixEvent.stopPropagation =function () { this.cancelBubble = true;
   };
```

fixEvent函数不是单独执行的，它必须有一个事件对象参数，而且只有事件发生时它才被执行！最好的方法是把它整合到addEvent函数的execEventHandles里面

```
   addEvent.execEventHandles = function (evt) {//遍历所有的事件处理函数并执行
       if (!this.__EventHandles) {return true;}
       evt = fixEvent(evt || window.event);//在这里对其进行标准化操作
       var fns = this.__EventHandles[evt.type]; for (var i=0;i< fns.length;i++) { if (fn
               fns[i].call(this,evt);//并且将其作为事件处理函数的第一个参数
               //这样在事件处理函数内部就可以使用统一的方法访问事件对象了
}
       }
   };
```

Load事件

使用JavaScript操纵DOM，必须等待DOM加载完毕才可以执行代码，但window.onload有个坏处，它非要等到页面中的所有图片及视频加载完毕才会触发load事件。结果就是一些本来应该在打开时隐藏起来的元素，由于网络延迟，在页面打开时仍然会出现，然后又会突然消失，让用户觉得莫名其妙。必须与这种丑陋的闪烁告别！

```
function addLoadEvent(fn) { var init = function() { if (arguments.callee.done) return;
            arguments.callee.done = true;
            fn.apply(document,arguments);
        }; //注册DOMContentLoaded事件，如果支持的话
        if (document.addEventListener) {
            document.addEventListener("DOMContentLoaded", init, false);
        } //但对于Safari，我们需要使用setInterval方法不断检测document.readyState
        //当为loaded或complete的时候表明DOM已经加载完毕
        if (/WebKit/i.test(navigator.userAgent)) { var _timer = setInterval(function() {
                    clearInterval(_timer);
                    init();
                }
            },10);
        } //对于IE则使用条件注释，并使用script标签的defer属性
         //IE中可以给script标签添加一个defer(延迟)属性，这样，标签中的脚本只有当DOM加载完毕后才执行
         /*@cc_on @*/
         /*@if (@_win32)
         document.write("<script id=\"__ie_onload\" defer=\"defer\" src=\"javascript:void
         var script = document.getElementById("__ie_onload");
         script.onreadystatechange = function() {
             if (this.readyState == "complete") {
                 init();
             }
         };
         /*@end @*/
         return true;
    }
```

javascript快速入门18--样式

修改元素外观方式

修改元素外观主要有下面3种方法：修改ID，修改className,修改元素的style属性

修改ID？会造成多么混乱的结果可想而知！

修改className确实是非常好的方法，我们甚至可以利用CSS层叠，通过修改body的className来修改整个页面的风格！前提是我们必须写特定的CSS！

```
//CSS代码
   body.redStyle { border:2px solid red;
   } body.redStyle * { color:red;
   } body.blueStyle { border:2px solid blue;
   } body.blueStyle * { color:blue;
   }
```

```
//JS代码
   document.body.className="redStyle";//切换为“红粉佳人”风格
   document.body.className="blueStyle";//切换为“蓝调情怀”风格
```

但修改className也并非那么容易，不要忘了className可以有多个的！所以不管添加，测试还是移除元素的className，都要小心，下面的函数可以造福人类！

```
function hasClassName(obj,cn) { return (new RegExp("\\b"+cn+"\\b")).test(obj.className);
   } function addClassName(obj,cn) { return obj.className += " " + cn;//第一次添加时，会多出
       //但不用担心，浏览会自动将其清除掉
} function delClassName(obj,cn) { return obj.className = obj.className.replace(new RegEx
   }
```

元素的style属性?见下面

Style属性

可以在元素的style属性上应用CSS规则，并且style属性上的规则优先级要高于样式表中的规则，因此，通过修改元素的style属性来修改元素的外观可能是最方便并且是最有效的方法了。

同其它HTML属性不同的是，元素的style属性是一个对象，CSS的属性名和属性值分别映射到了style对象的属性名和属性值，如定义对象的style="color:red;"，在JavaScript中访问时就可以这样访问：obj.style.color。但style属性也有一些需要注意的地方，比如CSS属性名中包含一些不能用作变量名的非法字符时，在JavaScript中访问时，会自动转换成驼峰命名式。

```
var oDiv=document.getElementById("oDiv");
   alert(oDiv.style.fontSize);//自动驼峰命名
   //如果要直接获取style属性中的所有CSS文本，则可以使用style对象的cssText属性
alert(oDiv.style.cssText); //对于CSS简写方式，各个浏览器返回结果出现分歧
alert(oDiv.style.border); //需要分别获取值
   alert(oDiv.style.borderLeftColor);//但会很麻烦
   //对于颜色，火狐总返回RGB表现形式,但设置时可以使用十六进制形式
   alert(oDiv.style.backgroundColor);//火狐会返回rgb(x,y,z)
```

但是元素的style属性仅仅提供了内联样式所定义的CSS规则，而不能反映CSS样式表中的规则

```
//CSS
   #oDiv {
       color:blue;
   }
   //HTML <div id="oDiv">Div</div> //JS
   alert(document.getElementById("oDiv").style.color);//空的
```

另外，要注意的是，修改元素的style属性时，必须将一个字符串赋给style对象的属性！

```
   oDiv.style.width = 120;//错误的，虽然在IE下有效果
   oDiv.style.fontSize="120%";//正确
```

获取实际应用在元素上的样式

对于获取元素的实际的CSS层叠最后的样式，IE与W3C DOM存在分歧：IE给对象提供了一个currentStyle属性，它的使用方式很像元素的style属性，但它返回的值是元素的实际样式，而不管样式是内联的还是在外部样式表中定义的！ W3C DOM则使用一个全局方法getComputedStyle，它的第一个参数为要检测的对象，第二个参数为null（在未来实现），将返回一个与元素的style也很相似的对象，但一是返回的对象是元素实际样式规则，二是它对于数值型属性总是返回像素值

```
   alert(oDiv.currentStyle.width);//IE，currentStyle保留原来定义在CSS中的单位
   alert(window.getComputedStyle(oDiv,null).width);//W3C DOM,并且总是返回计算后的像素值
   //另外，两种方式都不能获取那些CSS简写方式定义的，下面两个都会输出空
alert(oDiv.currentStyle.background);
   alert(getComputedStyle(oDiv,null).border);
```

Cross-Browser 获取元素实际样式的方式

```
function getStyle(obj,cn) { if (window.getComputedStyle) {//W3C DOM
           return window.getComputedStyle(obj,null)[cn];
       } else if (obj.currentStyle) {//IE
           return obj.currentStyle[cn];
       } return "";
   }
```

样式表

DOM也提供了访问外部样式表的方法，当然，也有无奈的兼容性问题！

向页面添加样式表

```
//使用添加节点的方法
   var lnk = document.createElement("link");
   lnk.type="text/css";
   lnk.rel="styleSheet";
   lnk.href="test.css"; var head = document.getElementsByTagName("head")[0];
   head.appendChild(lnk); //在IE下不能使用innerHTML向head中添加HTML代码的方式
   //另一种方法就是使用document.write
   document.write("");
```

访问样式表

也许修改个别元素的样式是十分简单的，但更改某条样式表规则（可以使所有相关元素都发生变化），则十分麻烦，并且这种技术只有Win平台上的IE与火狐才支持！但是访问样式表中的CSS规则仍然是有办法的！

```
   alert(document.styleSheets);//document对象的styleSheets属性是一个包含了所有的样式表的伪数组
   var sheets = docuemtn.styleSheets;
   alert(sheets.length);//length报告了样式表的个数
   //style标签出现一次或使用link标签链入CSS一次就算作一个样式表对象
   var sheet1 = sheets[0];//可以使用下标来访问
```

样式表对象的属性

- type，一般都为text/css
- href，link标签为其href属性，而style标签则为空
- id，所属标签的ID
- disabled，样式表是否启用，启用时为false
- cssText(仅IE)，样式表中规则的文本形式
- owningElement(IE)，ownerNode(W3C DOM)，返回引入样式表的那个标签
- rules(IE),cssRules(W3C)，对应样式表里所有规则的集合

Rule对象属性

- selectorText ，选择符
- style，与元素上的 style属性十分相似，可以读取和设置CSS规则 ，并且有cssText属性

修改样式表

通过styleSheets访问到的样式表对象具有一些方法来修改其中的样式表规则，但这些方法在各个浏览器中不一样。火狐支持W3C的insertRule()和deleteRule()方法。IE使用专有的addRule()和removeRule()方法。而其它浏览器则不支持任何一个。

```
var sheet = document.styleSheets[0];
   sheet.insertRule("body {color:blue;}",1);//W3C方法,第一个参数为CSS文本,第二个参数为位置,从0
   sheet.addRule("body","color:blue;",1);//IE方法,第一个参数为CSS选择符,第二个为CSS内容,第三个
   //而要删除某条规则，则只能通过下标
   sheet.removeRule(1);//IE
   sheet.deleteRule(1);//W3C
```

Cross-Browser 总结函数

```
function addCSS(sheet,selectorText,declarations,index) { if (sheet.insertRule) {
           sheet.insertRule(selectorText+" {"+declarations+"}",index);
       } else if (sheet.addRule) {
           sheet.addRule(selectorText,declarations,index);
       }
   } function delCSS(sheet,index) { if (sheet.deleteRule) {
           sheet.deleteRule(index);
       } else {
           sheet.removeRule(index);
       }
   }
```

javascript快速入门19--定位

元素尺寸

获取元素尺寸可以使用下面几种方式

- 元素的style属性width，height，但这些属性往往返回空值，因为它们只能返回使用行内style属性定义在元素上的样式
- 元素的currentStyle属性width，height（IE），getComputedStyle(obj,null)返回对象的width，height，这样可以获取元素的实际CSS定义的宽度和高度，但当元素没有使用CSS定义外观时，虽然元素仍然有大小（只要其中有字符或其它元素），这些属性的返回值是不确定的，如IE返回auto,而火狐则返回一个看似理想的值。
- 对象的clientWidth和clientHeight属性给出元素的可视部分的宽度和高度,当有滚动条时，只返回可见区域大小,对于块级元素,将返回元素的所设置的宽度和高度加上填充(padding)，这一点几乎所有浏览器都达成一致.但当块级元素并没有设置宽和高以及填充时,就出现不同了,谷歌浏览器和火狐浏览器的报告一致,IE报告都为0,而Opera则有所偏差.再将这两个属性应用到行内元素上,IE和火狐都报告为0,谷歌报告了一个看似理想的数字,而Opera竟会一个为正另一个负!
- 对象的offsetWidth和offsetHeight属性用来取得对象在页面中的实际所占区域大小(所设置的宽高加边框加填充，当有滚动条时还会算上滚动条),似乎这个属性对于设置了宽和高的块级元素几乎没有什么浏览器兼容问题,但不得不说的是火狐的一个BUG.火狐浏览器有个可将页面放大缩小的功能,当将页面缩小后,对象的offsetWidth和offsetHeight属性会发生细微的变化-变小几像素!尽管这对JS编程来讲几乎没影响,但似乎这个BUG也太明显了.这两个属性变非总是那么让人信任,当对象并没设置宽高或它是一个行内元素时,它的报告就显得相当复杂,不同浏览器都有自己一套标准(但是仍然可以肯定的是这两个属性报告的仍然是该元素占据的的空间大小,只不过会因字体和空格的默认大小不同而不同),最让人摸不着头脑的是Opera,对于一个body的子块级元素,当body和它自身没设置宽高时,Opera报告的它的宽度相当大,6千多像素!
- 还有就是scrollWidth和scrollWidth了,就目前来讲,对于一个没有滚动条和溢出的元素,其它浏览器对这个属性的报告还算有规律:对象的clientWidth+border=scrollWidth.对象的clientHeight+border=scrollHeight;只有IE报告有问题!它以元素中的内容为准,如果元素内没有其它内容,虽然IE并不会报0,但会报告一个非常小的值!再看看当元素有滚动条时怎么样吧!唯一值得高兴的是,它们对有滚动条的元素的clientWidth和clientHeight都报告一致(但仍有一点要注意,那就是火狐的一BUG,页面缩放功能带来的郁闷,而且这次变化非常大).而对于scrollWidth和scrollHeight真是五花八门:先说好的,尽管各不一样,但它们对scrollWidth不知为什么,相差不大,那么坏的就是,scrollHeight属性就相差太大了,没规律可循!(scrollWidth和scrollHeight属性返加对象内容的实际所需空间,当元素设置了overflow值为scroll或hidden之类时,scrollWidth和scrollHeight属性就派上用场了,可惜的是它问题太

多了)

综上所述，对于一个在CSS中定义了大小的块级元素，获取它的实际大小是很简单的，但对于没有定义宽度和高度，或是一行内元素时，则没有跨浏览器的解决方案可以获取它的实际大小！

窗口视口宽度和高度

对于窗口视口（视口指显示页面的那部分）的大小，Mozilla提供了window.innerWidth与window.innerHeight两个属性，而IE则没有相对应的属性，但可以使用document.documentElement.clientWidth与document.documentElement.clientHeight 两个属性来获取！另外，对于IE6之前的版本，则需要使用document.body的clientWidth与clientHeight属性！

```
//获取视口大小，依次为火狐，IE6及IE6以上的版本，IE6以下的版本
   var w = window.innerWidth || document.documentElement.clientWidth || document.body.cl
   //而同时其实火狐也支持通过document.documentElement的clientWidth,clientHeight属性获取视口大小
   //完全可以不做任何浏览器检测
   w = document.documentElement.clientWidth;
   h = document.documentElement.clientHeight;
```

元素边框宽度

获取元素边框宽度可以使用下面几种方式

- 元素style属性的borderWidth，但同样它只能返回使用HTML行内style属性定义的边框样式
- 使用currentStyle属性或getComputedStyle方法，可以获取元素CSS定义的实际边框样式，如果CSS没有定义元素的边框，一般的元素是没有边框的，但部分元素，如表单控件，仍然具有默认的边框宽度！对于此类情况，也是不能依靠这种方法来获取元素的边框大小的！
- 对象的clientLeft和clientTop属性。不能不说clientLeft和clientTop的名字起的太奇怪了,事实上它们的名字更应该这样:clientBorderLeftWidth和clientBorderTopWidth(难道是太长了的原因?);更奇怪的是,它们只能取得设置在元素上的左边框和上边框的粗细,而没有返回右边和下边的边框宽度的属性!另外,对于文档根元素(documentElement),在IE中它有默认的两像素边框宽度,而其它浏览器中报告为0.

元素坐标

获取元素坐标方法：

- 元素style属性的left,top属性，不但这种方法仅适用于使用HTML style属性声明样式的元

素，并且只有当元素使用定位（position设置为relative,absolute,fixed,但不包含static）时，才会存在这些值

- 元素currentStyle属性或getComputedStyle方法返回的Style对象的left,top属性，但这种方法仅对采用定位的元素有作用
- 元素的offsetLeft与offsetTop属性返回元素在页面中相对于父元素的坐标。一般对于进行了定位的元素（即position为不是static的值或没有设置），这两个属性的返回值为CSS中定义的元素的left,top值，当其自身有边距时（margin），还会加上边距。而对于没有采用定位的元素则显得比较复杂，我们只能考虑下设置了宽度和高度的块级元素，因为没有设置宽高，及行内元素，没有办法获取它的宽高，即使能获取它的left，top值也显得无意义了。对于没有采用定位的块级元素，offsetLeft与offsetTop属性将返回其自身的margin+父元素的padding。元素还有个offsetParent属性返回元素的相对定位的父元素，当使用定位时，各个浏览器一致，并且和CSS里设置的吻合，当不使用定位时，父元素是WHO成了问题，各个浏览器认识不一样，IE报告为其父节点，而其它浏览器则报告为body，当然，这次IE正确了。另外，对于表格中的一些元素，不应对其进行定位！

具有滚动条时的定位

scrollLeft和scrollTop,它们用来获得那些具有滚动条的元素滚动条滚动的距离,而没有滚动条的元素,它总返回0.可以这样认为,这两个属性报告了有滚动条元素中未显示的左一部分的宽(scrollLeft)和上一部分的高(scrollTop).而对于页面的滚动条，则取documentElement的scrollLeft与scrollTop属性，但是对于谷歌浏览器，它会将页面的滚动条视为document.body的！

Event对象与定位相关的属性

clientX与clientY返回事件发生时鼠标在视口中的坐标；offsetX与offsetY返回事件发生时鼠标相对于目标对象的坐标，以目标对象右上角为坐标原点，而这两个属性的W3C DOM版本则为layerX与layerY；pageX与pageY返回事件发生时鼠标相对于页面的坐标，虽然这个属性IE不支持，但仍然有补救的余地！

```
//不要在每个事件处理函数中进行判断，而要善于利用之前的fixEvent函数！
   function fixEvent(evt) { if (!evt.target) { //函数中已有的部分
           evt.pageX = evt.clientX+document.documentElement.scrollLeft;
           evt.pageY = evt.clientY+document.documentElement.scrollTop; //可以将事件发生时鼠
           evt.layerX = evt.offsetX;
           evt.layerY = evt.offsetY;
       } return evt;
   }
```

拖动

最简单的拖动脚本——拖动的基本原理

```
window.onload = function () { var oDiv = document.getElementById("oDiv");//oDiv必须使用
    oDiv.onmousedown = drag; function drag(evt) {
        evt = evt || window.event; this.onmouseup = drop; this.onmousemove = moveDiv;
            x:evt.offsetX || evt.layerX,
            y:evt.offsetY || evt.layerY
        };
    } function moveDiv(evt) {
        evt = evt || window.event; this.style.left = evt.clientX-this.offset.x+"px";
    } function drop(evt) { this.onmouseup = null; this.onmousemove = null;
    }
};
```

javascript快速入门20--Cookie

Cookie 基础知识

我们已经知道，在 document 对象中有一个 cookie 属性。但是 Cookie 又是什么？“某些 Web 站点在您的硬盘上用很小的文本文件存储了一些信息，这些文件就称为 Cookie。”—— MSIE 帮助。一般来说，Cookies 是 CGI 或类似，比 HTML 高级的文件、程序等创建的，但是 JavaScript 也提供了对 Cookies 的很全面的访问权利。

在继续之前，我们先要学一学 Cookie 的基本知识。

每个 Cookie 都是这样的：cookie名=cookie值；cookie本身仅仅是一个字符串，是一组名值对；多组名值对用分号加空格分隔！

"cookie名"的限制与 JavaScript 的命名限制大同小异，少了“不能用 JavaScript 关键字”，多了“只能用可以用在 URL 编码中的字符”。后者比较难懂，但是只要你只用字母和数字命名，就完全没有问题了。“值”的要求也是“只能用可以用在 URL 编码中的字符”。

每个 Cookie 都有失效日期，一旦电脑的时钟过了失效日期，这个 Cookie 就会被删掉。我们不能直接删掉一个 Cookie，但是可以用设定失效日期早于现在时刻的方法来间接删掉它。

每个网页，或者说每个站点，都有它自己的 Cookies，这些 Cookies 只能由这个站点下的网页来访问，来自其他站点或同一站点下未经授权的区域的网页，是不能访问的。每一“组”Cookies 有规定的总大小（大约 2KB 每“组”），一超过最大总大小，则最早失效的 Cookie 先被删除，来让新的 Cookie“安家”。

访问Cookie

```
    document.write(document.cookie);//输出类似"name1=value1; name2=value2; name3=value3"的
    document.write(typeof document.cookie);//cookie仅仅是个字符串
```

但这样获取到的是一堆混乱的字符串，必须对其进行处理才能知道它的含义！在类似ASP或PHP这样的服务器端脚本中，往往设置cookie十分简单

```
//ASP
    response.cookies("cookieName")="cookieValue"
    //PHP
    setcookie("cookieName","cookieValue");
```

解析Cookie名值对

方案一：直接截取字符串

```
function getCookie(cookieName) { var start = document.cookie.indexOf(cookieName+"="); if
        start = start+cookieName.length+1; var end = document.cookie.indexOf(";",start);
    }
```

方案二：将Cookie拆分为数组，通过遍历取得

```
function getCookie(cookieName) { var cookies=document.cookie.split("; ");//一个分号加一个空
        if (!cookies.length) {return "";} var pair=["",""]; for (var i=0;i< cookies.lengt
            pair = cookies[i].split("=");//以赋值号分隔,第一位是Cookie名,第二位是Cookie值
            if (pair[0]==cookieName) { break;
            }
        } return pair[1];
    }
```

方案三：使用正则表达式解析

```
function getCookie(cookieName) { var re = new RegExp("\\b"+cookieName+"=([^;]*)\\b"); va
    }
```

设置Cookie

一个Cookie包含以下信息：

- Cookie名称，Cookie名称必须使用只能用在URL中的字符，一般用字母及数字
- Cookie值，Cookie值同样也只能使用可以用在URL中的字符，一般需要在设置Cookie值时对其使用encodeURI方法进行转义
- Expires，过期日期，一个GMT格式的时间，当过了这个日期之后，浏览器就会将这个Cookie删除掉，当不设置这个的时候，Cookie在浏览器关闭后消失
- Path，一个路径，在这个路径下面的页面才可以访问该Cookie，一般设为“/”，以表示同一个站点的所有页面都可以访问这个Cookie
- Domain，子域，指定在该子域下才可以访问Cookie，例如要让Cookie在bbs.x-life.com下可以访问，但在news.x-life.com下不能访问，则可将domain设置成bbs.x-life.com
- Secure，安全性，指定Cookie是否只能通过https协议访问，一般的Cookie使用HTTP协议既可访问，如果设置了Secure（没有值），则只有当使用https协议连接时cookie才可以被页面访问

注意：**Cookie**安全机制要求站点页面只能访问本站点的**Cookie**，不能访问其它站点的**Cookie**。同时，最好在设置**Cookie**时使用**encodeURI**对象进行**URI**编码，在取出**Cookie**时再使用**decodeURI**对其进行解码！

设置一个完整Cookie示例

```
var expires = new Date();
   expires.setMonth(expires.getMonth()+1);//一个月后Cookie失效
   document.cookie = "userName="+encodeURI("用户名")+"; expires="+ expires.toGMTString()+
```

每次设置document.cookie值时如果该Cookie名称并不存在，则新增一个Cookie，如果已经存在，则修改以前的值！

```
   document.cookie ="a=1";//新增一个名称为a的Cookie
   document.cookie = "b=2";//新增一个名称为b的Cookie，原来的Cookie安然无恙
   document.cookie = "a=3";//将原来的名称为a的Cookie值修改为3
```

setCookie函数

```
function setCookie(name,value,expires,domain,secure) { var str = name+"="+encodeURI(valu
       if (expires) {
           str += "; expires="+expires.toGMTString();
       } if (path) {
           str += "; path="+path;
       } if (domain) {
           str += "; domain="+domain;
       } if (secure) {
           str += "; secure";
       }
       document.cookie = str;
   }
```

删除Cookie

没有删除Cookie的直接的方法，但可以变通一下来删除Cookie！

```
function delCookie(cookieName) { var expires = new Date();
       expires.setTime(expires.getTime()-1);//将expires设为一个过去的日期，浏览器会自动删除它
       document.cookie = cookieName+"=; expires="+expires.toGMTString();
   }
```

javascript快速入门21--DOM总结

跨浏览器开发

市场上的浏览器种类多的不计其数，它们的解释引擎各不相同，期待所有浏览器都一致的支持JavaScript,CSS,DOM,那要等到不知什么时候，然而开发者不能干等着那天。历史上已经有不少方法来解决浏览器兼容问题了，主要分为两种：1.userAgent字符串检测，2.对象检测；当然，也不能考虑所有的浏览器，我们需要按照客户需求来，如果可以确信浏览网站的用户都使用或大部分使用IE浏览器，那么你大可放心的使用IE专有的那些丰富的扩展，当然，一旦用户开始转向另一个浏览，那么痛苦的日子便开始了。下面是市场上的主流浏览器列表：

- Internet Explorer
- Mozilla Firefox
- Google Chrome
- Opera
- Safari

注意，浏览器总是不断更新，我们不但要为多种浏览器作兼容处理，还要对同一浏览器多个版本作兼容处理。比如IE浏览器，其6.0版本和7.0版本都很流行，因为微软IE随着操作系统绑定安装（之前也是同步发行，微软平均每两年推出一款个人桌面，同样IE也每两年更新一次；直到现在，由于火狐的流行，IE工作组才加快IE的更新），所以更新的较慢，6.0版和7.0版有很大差别。

市场上还存在一些其它浏览器，但由于它们都是使用的上面所列浏览器的核心，或与上面浏览器使用了相同的解释引擎，所以无需多作考虑。下面是主流的浏览器解释引擎列表：

1. Trident

   Trident （又称为MSHTML），是微软的窗口操作系统（Windows）搭载的网页浏览器—Internet Explorer的排版引擎的名称，它的第一个版本随着1997年10月Internet Explorer第四版释出，之后不断的加入新的技术并随着新版本的Internet Explorer释出。在未来最新的Internet Explorer第七版中，微软将对Trident排版引擎做了的重大的变动，除了加入新的技术之外，并增加对网页标准的支持。尽管这些变动已经在相当大的程度上落后了其它的排版引擎。使用该引擎的主要浏览器：IE，TheWorld，MiniIE，Maxthon，腾讯TT浏览器。事实上，这些浏览器是直接使用了IE核心，因为其userAgent字符串中返回的信息与IE是一模一样的！

2. Gecko

壁虎，英文为"Gecko"。Gecko是由Mozilla基金会开发的布局引擎的名字。它原本叫作NGLayout。Gecko的作用是读取诸如HTML、CSS、XUL和JavaScript等的网页内容，并呈现到用户屏幕或打印出来。Gecko已经被许多应用程序所使用，包括若干浏览器，例如Firefox、Mozilla Suite、Camino,Seamonkey等等

3. Presto

Presto是一个由Opera Software开发的浏览器排版引擎，供Opera 7.0及以上使用。Presto取代了旧版Opera 4至6版本使用的Elektra排版引擎，包括加入动态功能，例如网页或其部分可随着DOM及Script语法的事件而重新排版。Presto在推出后不断有更新版本推出，使不少错误得以修正，以及阅读Javascript效能得以最佳化，并成为速度最快的引擎。

4. KHTML

是HTML网页排版引擎之一，由KDE所开发。KDE系统自KDE2版起，在档案及网页浏览器使用了KHTML引擎。该引擎以C++编程语言所写，并以LGPL授权，支援大多数网页浏览标准。由于微软的Internet Explorer的占有率相当高，不少以FrontPage制作的网页均包含只有IE才能读取的非标准语法，为了使KHTML引擎可呈现的网页达到最多，部分IE专属的语法也一并支援。目前使用KHTML的浏览器有Safari和Google Chrome。而KHTML也产生了许多衍生品，如：WebKit,WebCore引擎

利用userAgent检测

下面是各大浏览器使用弹窗显示的userAgent字符串

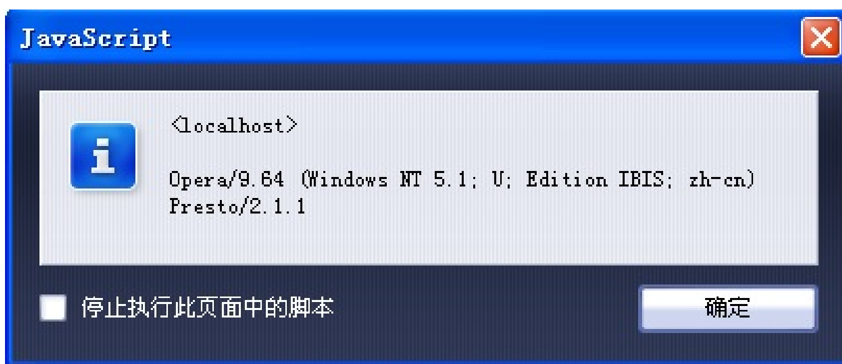
IE浏览器：Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727)



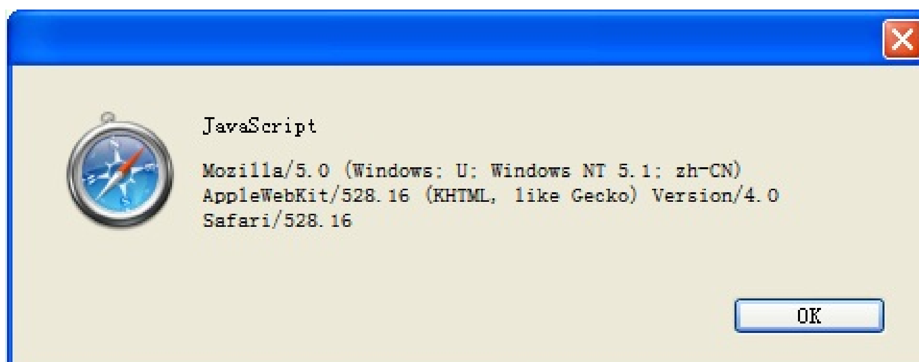
火狐浏览器：Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN; rv:1.9.0.4) Gecko/2008102920 Firefox/3.0.4



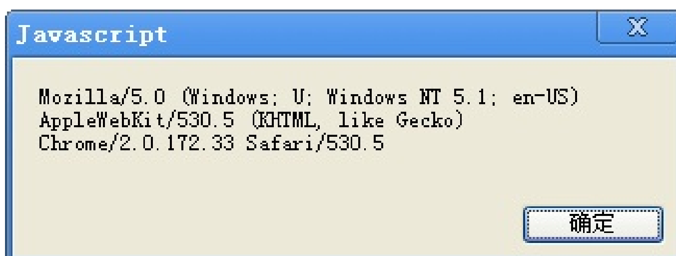
Opera浏览器：Opera/9.64 (Windows NT 5.1; U; Edition IBIS; zh-cn) Presto/2.1.1



Safari浏览器：Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN) AppleWebKit/528.16 (KHTML, like Gecko) Version/4.0 Safari/528.16



Google Chrome浏览器：Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/530.5 (KHTML, like Gecko) Chrome/2.0.172.33 Safari/530.5



可以使用下面的代码进行浏览器检测

```
var Browser = {
        isIE:navigator.userAgent.indexOf("MSIE")!=-1,
        isFF:navigator.userAgent.indexOf("Firefox")!=-1,
        isOpera:navigator.userAgent.indexOf("Opera")!=-1,
        isSafari:navigator.userAgent.indexOf("Safari")!=-1 };
```

但这样做并不是万无一失的，一个特例便是Opera可以使用userAgent伪装自己。下面是伪装成IE的userAgent：Mozilla/5.0 (Windows NT 5.1; U; Edition IBIS; zh-cn; rv:1.8.1) Gecko/20061208 Firefox/2.0.0 Opera 9.64；在完全伪装的情况下，最后的“Opera 9.64”这个字符串也不会出现，但Opera也有特殊的识别自身的方法，它会自动声明一个opera全局变量！

不但如此，我们的检测还忽略了一点，就是那些使用相同引擎而品牌不同的浏览器，所以，直接检测浏览器是没有必要的，检测浏览器的解释引擎才是有必要的！

```
var Browser = {
      isIE:navigator.userAgent.indexOf("MSIE")>-1 && !window.opera,
      isGecko:navigator.userAgent.indexOf("Gecko")>-1 && !window.opera && navigator.use
      isKHTML:navigator.userAgent.indexOf("KHTML")>-1,
      isOpera:navigator.userAgent.indexOf("Opera")>-1 };
```

对象检测

浏览器检测就到此结束了，下面应该讲一下对象检测！对象检测其实是比浏览器检测更加有效更加科学方法，而且我们之前一直在使用！

```
function addEvent(obj,evtype,bubbles) { if (obj.addEventListener) {....} else if (obj.at
   }
```

对象检测避免了浏览器引擎的多样性，即当我们需要某种功能时，我们直接检测浏览器是否支持该功能，而不用管浏览器是什么牌子的！

什么时候该使用浏览器检测？什么时候该使用对象检测？

答案是能使用对象检测时总该使用对象检测，只有当必须对浏览器进行识别或无法使用对象检测时才进行userAgent判断

```
//一段用于将当前页面添加到用户收藏夹的代码,两个不同的版本
   window.external.addFavorite(location,"收藏页面");//IE
   window.sidebar.addPanel("收藏页面",location,"");//火狐
   //由于在火狐下window也具有external属性，并且在IE下判断window.external.addFavorite会出错
   if (window.external.addFavorite) {...}//代码在IE下会出错
   //可以使用浏览器检测，避免意外
   if (Browser.isIE) {
       window.external.addFavorite(location,"收藏页面");//IE
   } else if (Browser.isGecko) {
       window.sidebar.addPanel("收藏页面",location,"");//火狐及其它相同引擎的浏览器
   }
```

当你的脚本和其它脚本一起工作时（尤其和那些有问题的脚本），有时候需要同时使用对象检测与浏览器检测

```
//对于window.innerWidth这些属性,可能有些脚本会创建一个兼容多个浏览器的同名属性
   if (isIE) {//它聪明的使用了浏览器检测
       window.onresize = function () {
           window.innerWidth =document.documentElement.clientWidth;
           window.innerHeight = document.documentElement.clientHeight;
       };
       window.onresize();
   } //然而我们的脚本对其进行检测时就麻烦了
   if (window.innerWidth) {//仅当在FF下时（FF支持position:fixed）才这样做
       obj.style.position="fixed";
       obj.style.right=window.innerWidth/2+"px";
       obj.style.bottom=window.innerWidth/2+"px";
....
   }
```

当然，使用浏览器检测始终是困难重重的，而对于测试文档是否支持某种特性，使用对象检测可能是最有效的，还有另一种方法可以直接测试浏览器是否支持某标准！

测试与DOM标准的一致性

document对象有个implementation属性，该属性只有一个方法hasFeature(),用来测试浏览器是否支持某个DOM标准！

```
//测试是否支持XML DOM 1.0
   var supportXML = document.implementation.hasFeature("XML", "1.0");
```

特 征	支持的版本	描 述
Core	1.0, 2.0, 3.0	基本的DOM，给予了用层次树来表示文档的能力
XML	1.0，2.0，3.0	核心的XML扩展，增加了对CDATA Section、处理指令和实体的支持
HTML	1.0，2.0	XML的HTML扩展，增加了对HTML特定元素和实体的支持
Views	2.0	基于特定样式完成对文档的格式化
StyleSheets	2.0	为文档关联样式表
CSS	2.0	支持级联样式表1（CSS Level 1）
CSS2	2.0	支持级联样式表2（CSS Level 2）
Events	2.0	通用DOM事件
UIEvents	2.0	用户界面事件
MouseEvents	2.0	由鼠标引起的事件（点击、鼠标经过，等等）
MutationEvents	2.0	当DOM树发生改变时引发的事件
HTMLEvents	2.0	HTML 4.01的事件
Range	2.0	操作DOM树中某个特定范围的对象和方法
Traversal	2.0	遍历DOM树的方法
LS	3.0	在文件和DOM树之间同步地载入和存储
LS-Async	3.0	在文件和DOM树之间异步地载入和存储
Validation	3.0	用于修改DOM树之后仍然保持其有效性的方法

尽管这个相当方便，但是，使用implementation.hasFeature()有其明显的缺陷——决定DOM实现是否对DOM标准的不同的部分相一致的，正是去进行实现的人（或公司）。要让这个方法对于任何值都返回true，那是很简单的，但这并不一定表示这个DOM实现真的和所有的标

准都一致了。目前为止，最精确的浏览器要数Mozilla，但它多少也有一些并不完全和DOM标准一致的地方，这个方法却返回为true。

错误处理

无尽的DOM兼容性问题，如果总是使用对象检测，就会带来无尽的if else之类的分支语句，而且检测某些对象的某些属性时还会引发错误（尤其是在IE下），因为一般的JavaScript对象都可以转换成布尔值，但浏览器内置的一些方法或对象并不是JavaScript创建了，它们不一定能够转换成布尔值！所以，使用错误处理语句不但避免了分支判断，而且可以很优雅的处理错误！

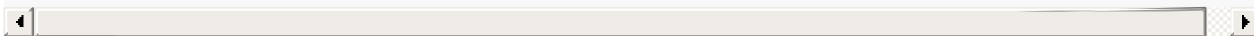
try {} catch(e) {} finally {}

```
function addFav(address,name) { try {
            window.sidebar.addPanel(name,address,"");//FF方法
            //在try语句中，如果脚本出错，会自动跳转到catch语句执行
        } catch(e) {//这里的e是必须的，是语法的一部分，它表示一个错误对象
            window.external.addFavorite(address,name);//IE
            //如果在这里还出现错误，浏览器就会将这个错误抛出
}
   }
```

Error对象

JavaScript内置了一个Error构造函数，可以创建一个错误对象，并可以使用throw语句手动抛出错误！

```
var err = new Error(); throw err; //在IE中，会在错误窗口中显示“未指明的错误”，而FF中则是空字符串
   //抛出错误后，脚本便会停止往下执行
   var message = "我抛出的错误！";//错误描述
   err = new Error(message); throw err;
```



错误对象的属性：message属性保存与错误对象相关的描述文字，number对象保存错误代码。（错误号是 32 位的值。高 16 位字是设备代码，而低字是实际的错误代码，不过错误代码对我们来说是没什么意义的）

```
try {
            undefined();//当try语句中脚本出现错误时，会自动抛出一个错误对象
        } catch(e) {//e是自动创建是局部变量，是Error的实例
            alert("错误信息是："+e.message+"\n"+"错误代码是："+e.number);
        } finally { //finally语句不管出现不出现错误，都将执行，一般用于出错时释放对象
        }
```

onerror事件处理

DOM还有个onerror事件处理，当页面载入出错，图象载入出错及页面脚本出错时会执行注册的事件处理函数，一个常用的方法便是，在事件处理函数中返回true可以阻止浏览器出现错误提示！

```
window.onerror = function () {
    alert("出错时你会看到我的！"); return true;
};
```

另外一个用法是，可以监测图像的载入，如果图像载入出错，可以重新载入图像（重新设置src属性），也可以利用这个方法来测试一下图像是否存在（比如检测用户输入的远程图像的URL是否有效）

尽管使用**try {} catch(e) {}**语句，及使用**onerror**事件处理可以处理脚本运行时错误，但它们是不能处理语法错误的！另外，使用错误处理语句并不是为了隐匿错误，而只是为了不干扰脚本的继续执行！

记录错误

处理了这么多错误后，记录错误这样的事情其实已经做过很多次了，主要有下面几种方法

- 使用alert语句，好处是可以让脚本暂停运行，坏处便是弹窗不那么好控制
- 使用document.write方法，这种方法自然避免了弹窗没法关闭的情况，但不好控制脚本运行！另一点便是，使用document.write会清空当前页面！
- 与document.write方法类似的是使用一个自建的控制台（比如一个DIV），然后将错误信息一条一条的添加进去，这种方法肯定比document.write好多了！
- 还有另一种方法便是使用浏览器内置的控制台，如在JavaScript里面可以调用Java控制台并向其中写入信息（前提是安装了JRE）

```
java.lang.System.out.println("错误信息！");//使用Java控制台
console.log("错误信息！");//使用火狐的控制台
opera.postError("错误信息！");//使用Opera的控制台
```

使用库提高开发效率

库，就是一些可以方便的应用到当前的开发体系中的代码资源，JavaScript库又被称之为JavaScript框架，它是由一些类和普通函数构成。使用库，可以使开发者不必关心程序的实现细节而只专心于业务逻辑。有很多流行的库，它们大都能够跨浏览器工作，并且采用了面向对象的良好编码方式。 事实上，库就是将之前我们写的那些可以跨浏览器运行的函数集中到一个JS文件中，以便能在所有页面都能够重复利用这些代码！当然，它们不是简简单单的将函数放到一起！下面是一些流行的库及其优缺点：

- Prototype

不要把它和JavaScript里面的prototype相混淆，Prototype是一个JS框架（官方称之为框架），可以说是最早也是最出名的JS框架了。 它提供了许多JS面向对象的扩展及DOM操作API，之前它一直由于缺乏API文档而备受诟病，但现在其文档已很充足。 它的优点显而易见，它只提供一些核心的，底层的功能，所以代码精简，体积较小，易学易用，但由于其只具有底层功能，往往需要协同其它UI库来运行！ 目前，基于Prototype的库已经有很多，著名的有集成Prototype库的RoR Ajax库，以及为Prototype提供许多视觉特效的Scriptaculous库。

- jQuery

  jQuery是一款同Prototype一样优秀js开发库，特别是对css和XPath的支持，使我们写js变得更加方便！如果你不是个js高手又想写出优秀的js效果，jQuery可以帮你达到目的！并且简洁的语法和高效率一直是jQuery追求的目标。（其官网标语：jQuery将改变您书写JavaScript的方式！）。 连YAHOO-UI都重用了很多jQuery的函数。支持插件是jQuery的另一大优点，可以无即的扩展其功能。其缺点便是内部结构复杂，代码较为晦涩，一般的新手根本无法看懂其源码。所以jQuery适合开发而不适合一个刚开始学习创建JS库的新手研究其源码。

- EXT-JS

  EXT-JS前身是YAHOO-UI，EXT-JS是具有CS风格的Web用户界面组件 能实现复杂的Layout布局，界面效果可以和BackBase（另一JS库）媲美，而且使用纯javascript代码开发。眞正的可编辑的表格Edit Grid，支持XML和Json数据类型，直接可以迁入grid。许多组件实现了对数据源的支持，例如动态的布局，可编辑的表格控件，动态加载的Tree 控件、动态拖拽效果等等。1.0 beta版开始同Jquery合作，推出基于jQuery的Ext 1.0，提供了更多有趣的功能。 对于一个喜欢JAVA的开发者来说，EXT-JS类似于java的结构，清晰明了，另一特点其可以实现华丽的令人震撼的WEB应用程序。当然，缺点也很严重，由于很多HTML及CSS代码都是EXT-JS自已创建的，所以其界面构造十分复杂，没有让我们自己写CSS的余地。

- Dojo

  Dojo是目前最为强大的工业级JS框架。它在自己的Wiki上给自己下了一个定义，dojo是一个用JavaScript编写的开源的DHTML工具箱。dojo很想做一个“大一统”的 工具箱，不仅仅是浏览器层面的，野心还是很大的。Dojo包括ajax, browser, event, widget等跨浏览器API，包括了JS本身的语言扩展，以及各个方面的工具类库，和比较完善的UI组件库，也被广泛 应用在很多项目中，他的UI组件的特点是通过给html标签增加tag的方式进行扩展，而不是通过写JS来生成，dojo的API模仿Java类库的组织 方式。 用dojo写Web OS可谓非常方便。dojo现在已经4.0了，dojo强大的地方在于界面和特效的封装，可以让开发者快速构建一些兼容标准的界面。 Dojo几乎集了成了上面的JS库的优点，其功能非同一般的强大，已得到了IBM和SUN的支持，但是自然的，其体积也十分大，总共有200多KB，另外，其语法也不如jQuery灵活，对JavaScript语言的增强也不如Prototype。

- Scriptaculous

Scriptaculous是基于prototype.js框架的JS效果。包含了6个js文件，不同的文件对应不同的js效果，所以说，如果底层用 prototype的话，做js效果用Scriptaculous那是再合适不过的了。优点便是基于Prototype，可以说是Prototype的插件，不同的效果用不同的JS文件分开存放，当然，依赖于Prototype也是其缺点。

- moo.fx

  moo.fx是一个超级轻量级的javascript特效库（3k）,能够与prototype.js框架一起使用。它非常快、易于使用、跨浏览器、符合标准，提供控制和修改任何HTML元素的CSS属性，包括颜色。它内置检查器能够防止用户通过多次或疯狂点击来破坏效果。moo.fx整体采用模块化设计，所以可以在它的基础上开发你需要的任何特效。轻量是其最大的优点，当然其缺点便是轻量级的，但是轻量级的JS库能有如此强大已经很不错了。

命名空间

在创建库之前，第一个要考虑的问题便是如何防止变量重名的问题！解决方案便是使用命名空间！命名空间是JAVA等语言中的一个概念，JavaScript并不支持命名空间，但由于JavaScript的灵活性，我们可以使用对象来模似命名空间！

```
var NameSpace={};
   NameSpace.fn1 =function () {};
   NameSpace.fn2 =function () {};
```

现在，fn1与fn2两个函数就同属于NameSpace命名空间中了，它们不会也全局中的fn1或fn2冲突，当然，在使用的时候必需加上NameSpace前缀。这样使用命名空间并不是最简单的，因为我们不得不在任何地方调用这些函数的时候都加上NameSpace前缀。下面是使用JavaScript闭包来解决的方案：

```
var NameSpace={};
   (function () { function fn1() {
       } function fn2() {
           fn1();//在这里调用fn1不需要加NameSpace前缀
}
       NameSpace.fn1=fn1;//将其添加为全局变量NameSpace的属性，以便能在函数外访问
       NameSpace.fn2=fn2;
   })();
```

javascript快速入门22--Ajax简介

Ajax是什么?

首先,Ajax是什么?一个很酷的新兴词汇!仅仅是某种早就有了的技术的一种新说法而已! Ajax是指一种创建交互式网页应用的网页开发技术。要谈到网页应用程序,则必须从WEB的历史来讲:

1.开始的Internet,仅仅是科学家们用来交换研究论文,及一些大学在上面发布一些课程信息的工具,那个时候网页与一幅户外广告没多大区别(相反户外广告才能起来广告的作用).那个时候,只有少部分的公司具有公司网站,而它们的公司网站仅仅是在首页上放置一些联系信息或一些静态的文档!

2.当Windows出现后(尽管Windows仅仅是给早就有了的操作系统加个外套而已,但这确实上一大进步),及个人电脑的流行,WEB也开始从学院走向群众,人们无法再忍受静态网页的一成不变,于是CGI(Common Gateway Interface)诞生了! CGI其实是用C或Perl编写的程序,当用户请求某个页面时,CGI程序会自动执行,CGI程序可以访问数据库,返回HTML页面.那个时候就可以通过CGI来创建一个在线商城了.然而CGI有很多缺点:首先是其编写很复杂,往往编写CGI的是一些专业的程序员,他们只会关心一些算法问题,而不会理HTML页面是否漂亮! 另一点,由于CGI是经过编译后的程序,虽然作为独立程序运行时效率会很高,但也很危险,因为CGI程序可以访问服务器的系统里的其它与WEB无关的程序及创建文件,虽然一般情况下CGI程序不会这样做,但如果恶意用户将CGI程序放到服务器,那么它就可以为所欲为了!尽管存在这些缺陷，到如今CGI仍在使用。

3.很多人都知道Sun,知道因特网流行的编程语言JAVA.如上所说CGI具有许多缺点,JAVA便是来弥补这些缺点的.由于Netscape的Navigator支持Java，动态Web页面掀开了新的一页：applet时代到来了。Applet与CGI不同,它是运行在客户端的,Applet就是嵌入在Web页面上的小应用程序. 只要用户使用支持Java的浏览器，就可以在浏览器的Java虚拟机（Java Virtual Machine，JVM）中运行applet。尽管applet可以做很多事情，但它也存在一些限制：通常不允许它读写文件系统，它也不能加载本地库，而且可能无法启动客户端上的程序。除了这些限制外，applet是在一个沙箱安全模型中运行的，这是为了有助于防止用户运行恶意代码。JAVA最先就是因为Applet出名的,而很多人学JAVA也是从Applet开始的.然而,Applet好景不长,一是由于一个Applet本身加载要很长时间,另一方面,由于更流行的MS的IE开始不支持Applet,它就只好没落了.

4.与此同时，Netscape创建了一种脚本语言，并最终命名为JavaScript（建立原型时叫做Mocha，正式发布之前曾经改名为LiveWire和LiveScript，不过最后终于确定为JavaScript）。设计JavaScript是为了让不太熟悉Java的Web设计人员和程序员能够更轻松地开发applet（当然，Microsoft也推出了与JavaScript相对应的脚本语言，称为VBScript）。 当

然,最初JavaScript是很失败的,由于各个浏览器相互不兼容(然而它们都提供了弹窗,那些烦人的alert),又因为缺乏开发工具,JavaScript很受非议.但尽管如此，JavaScript仍然是一种创建动态Web应用的强大方法。

5.在Java问世一年左右，Sun引入了servlet。现在Java代码不用再像applet那样在客户端浏览器中运行了，它可以在你控制的一个应用服务器上运行。这样，开发人员就能充分利用现有的业务应用，而且，如果需要升级为最新的Java版本，只需要考虑服务器就行了。Java推崇“一次编写，到处运行”，这一点使得开发人员可以选择最先进的应用服务器和服务器环境，这也是这种新技术的另一个优点。servlet还可以取代CGI脚本。 当然,这个时候的servlet仍然比CGI简单不了多少.MS吸取了Sun的教训,推出了ASP,Sun也很快作出了回应,推出了JSP.JSP和ASP的设计目的都是为了将业务处理与页面外观相分离，从这个意义上讲，二者是相似的。虽然存在一些技术上的差别（Sun也从Microsoft那里学到了教训），但它们有一个最大的共同点，即Web设计人员能够专心设计页面外观，而软件开发人员可以专心开发业务逻辑。ASP与JSP都没有垄断服务器脚本市场,因为还有其它优秀的服务器脚本,如PHP,ColdFusion及Ruby!

6.当WEB进化到这里的时候,动态的网站已经很多了.但人们对动态的定义一直很模糊,比如说很多人以为动态是指动画!其实这也没什么可笑的,正是这一理念,将动态WEB从服务器端动态生成HTML进化为富客户端应用程序(当然不是动画). 富客户端应用程序(Rich Internet Applications,富因特网应用程序,RIA)的提出解决了长久以来的"客户体验"问题,用户在使用Windows时已经习惯于那些各色各样的桌面应用程序,而死板的HTML页面往往只能提供一些文档. RIA的出现,目标就是能使WEB页面能像桌面应用程序一样具有很高的交互性及响应率.其实Sun推出的Applet就是一个RIA,当然MS也有其产品,最近推出的SilverLight.并不只有这两家,还有Adobe Flash(它不是做动画的吗?对的,但其不但可以用来做动画,还可以创建一些其它的RIA),利用Flash，设计人员可以创建令人惊叹的动态应用。公司可以在Web上发布高度交互性的应用，几乎与胖客户应用相差无几。不同于applet、servlet和CGI脚本，Flash不需要编程技巧，很容易上手。 像许多解决方案一样，Flash需要客户端软件,由于此限制,很多网站上就多出了"跳过此页"的链接.

7.主角出场了(其实已经低调出场过一次了),曾经的JavaScript,以及其带来的DHTML,开始了新的历程. 当Microsoft和Netscape发布其各自浏览器的第4版时，Web开发人员有了一个新的选择：动态HTML（Dynamic HTML，DHTML）。与有些人想像的不同DHTML不是一个W3C标准，它更像是一种营销手段。实际上，DHTML结合了HTML、层叠样式表（Cascading Style Sheets，CSS）、JavaScript和DOM。这些技术的结合使得开发人员可以动态地修改Web页面的内容和结构。 最初DHTML的反响很好。不过，它需要的浏览器版本还没有得到广泛采用。尽管IE和Netscape都支持DHTML，但是它们的实现大相径庭，这要求开发人员必须知道他们的客户使用什么浏览器。而这通常意味着需要大量代码来检查浏览器的类型和版本，这就进一步增加了开发的开销。有些人对于尝试这种方法很是迟疑，因为DHTML还没有一个官方的标准。 当DHTML渐渐退出视野之后,我们的JavaScript并没有没落,由于W3C标准的不断推进,给JS带来了福音,现在编写跨浏览器的代码并不像当初那样困难了(尽管也存在一些问题).另外,XML与异步通信(XMLHttpRequest)在WEB上的的流行,浏览器对支持也越来越好,也使得

JS可以大展其身手.当你使用JS操纵DOM的时候,就发现实现动态WEB应用程序已经不再有多遥远. 现在又有了XML(数据库)与异步通信的支持,可以使得应用程序在加载完成后,无需跳转,就可以返回给用户所有的内容了!

Ajax是多种技术的并称

Ajax是Asynchronous JavaScript and XML（异步JavaScript和XML），它其实包含了很多技术,主要是下面所列的:

- ECMAScript,为什么不是JavaScript?因为JavaScript本身与Ajax一样,包含的也太多了!
- DOM及相关内容:CSS,XHTML....
- XML,及XML的一些扩展语言:XSL,SVG,XUL,XAML.....XML的扩展太多了
- XMLHttpRequest对象,浏览器提供的一个可以用于异步通信的JavaScrip对象
- 服务器脚本的支持,没有服务器脚本在后台工作,也是没有办法实现"动态"的,之后就知道了...

关于XMLHttpRequest对象

XMLHttpRequest对象其实最早是由MS提出来的,并在IE5中就提供了支持,当时,在IE5里它是一个ActiveXObject. 原先，XHR对象只在IE中得到支持（因此限制了它的使用），但是从Mozilla 1.0和Safari 1.2开始，对XHR对象的支持开始普及。这个很少使用的对象和相关的基本概念甚至已经出现在W3C标准中：DOM Level 3 加载和保存规约（DOM Level 3 Load and Save Specification）。现在，特别是随着Google Maps、Google Suggest、Gmail、Flickr、Netflix和A9等应用变得越来越炙手可热，XHR也已经成为事实上的标准。 与前面提到的方法不同，Ajax在大多数现代浏览器中都能使用，而且不需要任何专门的软件或硬件。实际上，这种方法的一大优势就是开发人员不需要学习一种新的语言，也不必完全丢掉他们原先掌握的服务器端技术。Ajax是一种客户端方法，可以与J2EE、.NET、PHP、Ruby和CGI脚本交互，它并不关心服务器是什么。尽管存在一些很小的安全限制，你还是可以现在就开始使用Ajax，而且能充分利用你原有的知识。

早期的异步通信实现

虽然可以使用XHR对象来实现异步通信,但其实早期的开发人员曾经也尝试过使用隐藏帧等方法来实现异步通信!

```
//主页面中的JS代码
    function getPages(url) { var iframe =document.getElementById("hideIframe");//一个隐藏了
        iframe.src = url;//将帧的src设置为传入的url,就可以将那个页面在后台载入
}
    window.container = document.getElementById("oDiv");//加载内容的窗口
    getPages("test.php?param=value");//可以通过QS传递参数
    //在隐藏帧中加载的页面中JS代码
    window.onload = function () {//当帧加载完毕后修改父窗口中的内容
        parent.container.innerHTML = document.body.innerHTML;
    }
```

另外还有一种就是使用script标签

```
//HTML
    <script type="text/javascript" id="voidScript" src="void(0)"> </script>
    //JS
    function getScript(url) { var script = document.getElementById("voidScript");
        script.src =url;//这种方法必须加载JS脚本,并且脚本加载后就会执行
}
    getScript("test.php?userName=abc");
```

Ajax的问题

Ajax并不是每个网站都需要的,尽管它有诸多优点:如可与XHTML无缝集成,轻量,无需插件..但其缺点也有不少:依赖JavaScript,影响浏览器默认行为如后退按钮及收藏夹等.另外,它最大的优点也是它最大的缺点: 我们以前总是告诉用户，Web应用是以一种请求/响应模式完成操作的，用户也已经接受了这种思想。但是用了Ajax，就不再有这个限制。我们可以只修改页面的一部分，如果用户没想到这一点，他们会尝试狂点某按钮,或刷新页面,所以要通过一些方法来让用户知道页面正在"异步"与服务器交互!

javascript快速入门23--XHR—XMLHttpRequest对象

创建XMLHttpRequest对象

与之前众多DOM操作一样,创建XHR对象也具有兼容性问题:IE6及之前的版本使用ActiveXObject,IE7之后及其它浏览器使用XMLHttpRequest

不但IE6及其之前的版本将XHR作为一个ActiveXObject运行,而且还存在众多版本:一开始是Microsoft.XMLHTTP 之后变成Msxml2.XMLHTTP及更新版的Msxml3.XMLHTTP

```
function XHR() { var xhr; try {xhr = new XMLHttpRequest();} catch(e) { var IEXHRVers =["
            }
        } return xhr;
    }
```

使用XHR

```
var xhr = XHR(); //open方法 创建一个新的http请求，并指定此请求的方法、URL以及验证信息(用户名/密码)
    xhr.open("get","test.txt",true); /*第一个参数是请求方式,一般用get与post方法,与form标签的meth
    第二个参数是请求的URL
    第三个参数是请求是同步进行还是异步进行,true表示异步
    调用了open方法仅仅是传递了参数而已*/ xhr.send(null);//调用了send方法后才会发出请求
    //并且get方式发送请求时send参数是null
```

在服务器环境中执行上面的脚本,并且给一个php或asp脚本发送请求,会发现服务器端脚本其实会被执行

```
//PHP脚本
    $fp =fopen("a.txt","wb"); fwrite($fp,"PHP文件在后台执行了"); fclose($fp); echo "返回内容!
```

上面PHP脚本的返回内容不会直接在页面上显示出来,必需要用JS通过XHR对象接收

```
var xhr = XHR();
   xhr.open("get","test.php",true);
   xhr.onreadystatechange = callback;//在readystatechange事件上绑定一个函数
   //当接收到数据时,会调用readystatechange事件上的事件处理函数
   xhr.send(null);
   functon callback() { //在这里面没有使用this.readyState这是因为IE下面ActiveXObject的特殊性
       if (xhr.readyState == 4) {//readyState表示文档加载进度,4表示完毕
           alert(xhr.reponseText);//responseText属性用来取得返回的文本
}
   }
```

XHR对象参考

readyState属性 返回当前请求的状态

- 0 (未初始化) 对象已建立，但是尚未初始化（尚未调用open方法）
- 1 (初始化) 对象已建立，尚未调用send方法
- 2 (发送数据) send方法已调用，但是当前的状态及http头未知
- 3 (数据传送中) 已接收部分数据，因为响应及http头不全，这时通过responseBody和responseText获取部分数据会出现错误
- 4 (完成) 数据接收完毕,此时可以通过通过responseBody和responseText获取完整的回应数据

```
var xhr =XHR();
   alert(xhr.readyState);//0
   xhr.open("get","test.htm",true);
   alert(xhr.readyState);//1
   xhr.send(null);
   alert(xhr.readyState);//IE下会是4,而FF下会是2
   //可以通过readystatechange事件监听
   xhr = XHR();
   xhr.onreadystatechange = function () {
       alert(xhr.readyState);//FF下会依次是1,2,3,4但最后还会再来个1
       //IE下则是1,1,3,4
};
   xhr.open("get","test.txt",true);
   xhr.send(null);
```

从上面可以看到,对于readyState这个属性,各个浏览器看法也不一样,但其实我们只需要知道当状态为4的时候可以获取response就行了!

status 返回当前请求的http状态码

status属性返回当前请求的http状态码,此属性仅当数据发送并接收完毕后才可获取。完整的HTTP状态码如下:

- 100 Continue 初始的请求已经接受，客户应当继续发送请求的其余部分
- 101 Switching Protocols 服务器将遵从客户的请求转换到另外一种协议
- 200 OK 一切正常，对GET和POST请求的应答文档跟在后面。
- 201 Created 服务器已经创建了文档，Location头给出了它的URL。

- 202 Accepted 已经接受请求，但处理尚未完成。
- 203 Non-Authoritative Information 文档已经正常地返回，但一些应答头可能不正确，因为使用的是文档的拷贝
- 204 No Content 没有新文档，浏览器应该继续显示原来的文档。如果用户定期地刷新页面，而Servlet可以确定用户文档足够新，这个状态代码是很有用的
- 205 Reset Content 没有新的内容，但浏览器应该重置它所显示的内容。用来强制浏览器清除表单输入内容
- 206 Partial Content 客户发送了一个带有Range头的GET请求，服务器完成了它
- 300 Multiple Choices 客户请求的文档可以在多个位置找到，这些位置已经在返回的文档内列出。如果服务器要提出优先选择，则应该在Location应答头指明。
- 301 Moved Permanently 客户请求的文档在其他地方，新的URL在Location头中给出，浏览器应该自动地访问新的URL。
- 302 Found 类似于301，但新的URL应该被视为临时性的替代，而不是永久性的。
- 303 See Other 类似于301/302，不同之处在于，如果原来的请求是POST，Location头指定的重定向目标文档应该通过GET提取
- 304 Not Modified 客户端有缓冲的文档并发出了一个条件性的请求（一般是提供If-Modified-Since头表示客户只想比指定日期更新的文档）。服务器告诉客户，原来缓冲的文档还可以继续使用。
- 305 Use Proxy 客户请求的文档应该通过Location头所指明的代理服务器提取
- 307 Temporary Redirect 和302（Found）相同。许多浏览器会错误地响应302应答进行重定向，即使原来的请求是POST，即使它实际上只能在POST请求的应答是303时才能重定向。由于这个原因，HTTP 1.1新增了307，以便更加清除地区分几个状态代码：当出现303应答时，浏览器可以跟随重定向的GET和POST请求；如果是307应答，则浏览器只能跟随对GET请求的重定向。
- 400 Bad Request 请求出现语法错误。
- 401 Unauthorized 客户试图未经授权访问受密码保护的页面。应答中会包含一个WWW-Authenticate头，浏览器据此显示用户名字/密码对话框，然后在填写合适的Authorization头后再次发出请求。
- 403 Forbidden 资源不可用。
- 404 Not Found 无法找到指定位置的资源
- 405 Method Not Allowed 请求方法（GET、POST、HEAD、Delete、PUT、TRACE等）对指定的资源不适用。
- 406 Not Acceptable 指定的资源已经找到，但它的MIME类型和客户在Accpet头中所指定的不兼容
- 407 Proxy Authentication Required 类似于401，表示客户必须先经过代理服务器的授权。
- 408 Request Timeout 在服务器许可的等待时间内，客户一直没有发出任何请求。客户可以在以后重复同一请求。
- 409 Conflict 通常和PUT请求有关。由于请求和资源的当前状态相冲突，因此请求不能成功。

- 410 Gone 所请求的文档已经不再可用，而且服务器不知道应该重定向到哪一个地址。它和404的不同在于，返回407表示文档永久地离开了指定的位置，而404表示由于未知的原因文档不可用。
- 411 Length Required 服务器不能处理请求，除非客户发送一个Content-Length头
- 412 Precondition Failed 请求头中指定的一些前提条件失败
- 413 Request Entity Too Large 目标文档的大小超过服务器当前愿意处理的大小。如果服务器认为自己能够稍后再处理该请求，则应该提供一个Retry-After头
- 414 Request URI Too Long URI太长
- 416 Requested Range Not Satisfiable 服务器不能满足客户在请求中指定的Range头
- 500 Internal Server Error 服务器遇到了意料不到的情况，不能完成客户的请求
- 501 Not Implemented 服务器不支持实现请求所需要的功能。例如，客户发出了一个服务器不支持的PUT请求
- 502 Bad Gateway 服务器作为网关或者代理时，为了完成请求访问下一个服务器，但该服务器返回了非法的应答
- 503 Service Unavailable 服务器由于维护或者负载过重未能应答。例如，Servlet可能在数据库连接池已满的情况下返回503。服务器返回503时可以提供一个Retry-After头
- 504 Gateway Timeout 由作为代理或网关的服务器使用，表示不能及时地从远程服务器获得应答
- 505 HTTP Version Not Supported 服务器不支持请求中所指明的HTTP版本

事实上,我们只需要知道状态为200的时候(OK)才读取response就行了!

responseText与responseXML

responseText 将响应信息作为字符串返回 . XMLHTTP尝试将响应信息解码为Unicode字符串，XMLHTTP默认将响应数据的编码定为UTF-8，如果服务器返回的数据带BOM(byte-order mark)，XMLHTTP可以解码任何UCS-2 (big or little endian)或者UCS-4 数据。注意，如果服务器返回的是xml文档，此属性并不处理xml文档中的编码声明。你需要使用responseXML来处理。

responseXML 将响应信息格式化为Xml Document对象并返回 . 如果响应数据不是有效的XML文档，此属性本身不返回XMLDOMParseError，可以通过处理过的DOMDocument对象获取错误信息。

其它一些XHR对象的方法

abort 取消当前请求

getAllResponseHeaders 获取响应的所有http头 每个http头名称和值用冒号分割，并以\r\n结束。当send方法完成后才可调用该方法。

getResponseHeader 从响应信息中获取指定的http头 当send方法成功后才可调用该方法。如果服务器返回的文档类型为"text/xml", 则这句话xmlhttp.getResponseHeader("Content-Type");将返回字符串"text/xml"。可以使用getAllResponseHeaders方法获取完整的http头信息。

setRequestHeader 单独指定请求的某个http头 如果已经存在已此名称命名的http头，则覆盖之。此方法必须在open方法后调用。

请求方式

GET 请求

```
//JS
    var xhr = XHR();
    xhr.open("get","test.php?qs=true&userName=abc&pwd=123456",true);
    xhr.onreadystatechange = function () { if (xhr.readyState==4 && xhr.status ==200) {
            alert(xhr.responseText);
        }
    };
    xhr.send(null); //PHP
    print_r($_GET);
```

POST 请求

```
//JS
    var xhr = XHR();
    xhr.open("post","test.php",true);
    xhr.onreadystatechange = function () { if (xhr.readyState==4 && xhr.status ==200) {
            alert(xhr.responseText);
        }
    }; //比GET请求多了一步
    xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded"); //另外,数据
    xhr.send("qs=true&userName=abc&pwd=123456"); //PHP
    header("Cache-Control: no-cache, must-revalidate");//可以让浏览器不缓存结果
    print_r($_POST);
```

javascript快速入门24--XML基础

XML简介

XML代表Extensible Markup Language（eXtensible Markup Language的缩写，意为可扩展的标记语言）。XML 被设计用来传输和存储数据。XML是一套定义语义标记的规则，这些标记将文档分成许多部件并对这些部件加以标识。它也是元标记语言，即定义了用于定义其他与特定领域有关的、语义的、结构化的标记语言的句法语言。

XML特点

- 标记需要自己去创建!一个最大的特殊处(与HTML不同处),它可以创建标签,包括使用中文标签
- 语法更严格!标签必需成对嵌套,并且属性必须有属性值,属性值也必须包含在引号中
- XML仅仅用来存储数据,不包含与数据显示相关的样式信息(这部分信息可以通过CSS等语言来传递),XML是具有语义的

书写XML

一个完整的XML必须包含XML声明与一个文档根元素!XML声明必须包含版本信息,文档根元素可以是空元素,但一个XML文档只能有一个根元素!

```
<?xml version="1.0"?>
   <root />
```

注意**:XML**目前存在**1.0**与**1.1**两个版本**,**但浏览器只支持**1.0**版**. XML**声明的**version**属性是必须的**. XML**声明之前不能有任何东西**,**比如空格及注释**(XML**注释和**HTML**注释是一样的**).XML**文档只能且必须包含一个根元素**,**当元素是空元素时需要闭合

XML声明还有其它两个属性:standalone与encoding.encoding属性用来指明XML文档所使用的字符编码,如指定为gbk或gb2312则可以使用中文本,standalone属性指明XML文档是否独立不依赖于外部文档,默认值是yes,当使用外部XML DTD时需要将这个属性设为no

格式良好的XML与有效的XML

格式良好的XML(Well-formed XML)是指文档格式符合XML语法规范的XML,解释器在解释一个Not Well-formed XML的时会出现错误而停止!

一个遵守XML语法规则，并遵守相应DTD文件规范的XML文档称为有效的XML文档(Valid XML)。注意我们比较"Well-formed XML"和"Valid XML"，它们最大的差别在于一个完全遵守XML规范，一个则有自己的"文件类型定义(DTD)"。 将XML文档和它的DTD文件进行比较分析，看是否符合DTD规则的过程叫validation(确认)。这样的过程通常我们是通过一个名为parser的软件来处理的。

DTD——文档类型定义(Document Type Definition)

由于XML可以自定义标签,那么自然各人编写的标签不一样,这样同步数据便成了问题,因为其它人不知道某个标签应该怎么用,表示什么意思.DTD就是为了解决此问题的!

DTD是一种保证XML文档格式正确的有效方法，可以比较XML文档和DTD文件来看文档是否符合规范，元素和标签使用是否正确。一个DTD文档包含：元素的定义规则，元素间关系的定义规则，元素可使用的属性，可使用的实体或符号规则。

DTD分为内部DTD与外部DTD,内部DTD包含在XML文档中,外部DTD则通过URL引用.一个DTD文件是以.dtd结尾的文本文件

在XML中引入DTD DOCTYPE 文档类型声明

内部DTD,可以将standalone设置成yes.

```
<?xml version="1.0" standalone="yes"?>
   <!DOCTYPE root [
   <!ELEMENT root EMPTY> ]>
```

外部DTD,需要将standalone设成no

```
<?xml version="1.0" standalone="no"?>
   <!DOCTYPE root SYSTEM "http://www.test.org/test.dtd">
```

DOCTYPE分析

DTD声明始终以!DOCTYPE开头,空一格后跟着文档根元素的名称,如果是内部DTD,则再空一格出现[],在中括号中是文档类型定义的内容. 而对于外部DTD,则又分为私有DTD与公共DTD,私有DTD使用SYSTEM表示,接着是外部DTD的URL. 而公共DTD则使用PUBLIC,接着是DTD公共名称,接着是DTD的URL.下面是一些示例

公共DTD,DTD名称格式为"注册//组织//类型 标签//语言","注册"指示组织是否由国际标准化组织(ISO)注册,+表示是,-表示不是."组织"即组织名称,如:W3C; "类型"一般是DTD,"标签"是指定公开文本描述，即对所引用的公开文本的唯一描述性名称,后面可附带版本号。最后"语言"是DTD语言的ISO 639语言标识符,如:EN表示英文,ZH表示中文,在下面的地址有完整的ISO 639语言标识符列表http://ftp.ics.uci.edu/pub/ietf/http/related/iso639.txt

```
<!DOCTYPE root SYSTEM "http://www.test.org/test.dtd">
```

下面是XHTML 1.0 Transitional的DTD.以!DOCTYPE开始,html是文档根元素名称,PUBLIC表示是公共DTD,后面是DTD名称,以-开头表示是非ISO组织 组织名称是W3C,EN表示DTD语言是英语,最后是DTD的URL

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml/DTD/xhtml1-transitional.dtd">
```

注意:虽然DTD的文件URL可以使用相对URL也可以使用绝对URL,但推荐标准是使用绝对URL.另一方面,对于公共DTD,如果解释器能够识别其名称,则不去查看URL上的DTD文件

开始编写DTD

XML 文档构建模块

所有的 XML 文档（以及 HTML 文档）均由以下简单的构建模块构成：

- 元素，元素即所说的自定义标签,它是 XML 以及 HTML 文档的主要构建模块。
- 属性，属性可提供有关元素的额外信息。属性总是被置于某元素的开始标签中。属性总是以名称/值的形式成对出现的。
- 实体，实体是用来定义普通文本的变量。实体引用是对实体的引用。如HTML文档中的 即是一个实体引用当文档被 XML 解析器解析时，实体就会被展开。
- PCDATA，PCDATA 的意思是被解析的字符数据（parsed character data）。可把字符数据想象为 XML 元素的开始标签与结束标签之间的文本。PDATA 是会被解析器解析的文本。这些文本将被解析器检查实体以及标记。文本中的标签会被当作标记来处理，而实体会被展开。不过，被解析的字符数据不应当包含任何&、< 或者 > 字符；需要使用 &、< 以及 > 实体来分别替换它们。
- CDATA，CDATA 的意思是字符数据（character data）。CDATA 是不会被解析器解析的文本。在这些文本中的标签不会被当作标记来对待，其中的实体也不会被展开。

CDATA

PCDATA是指会被解析的字符串,这解析是指将其中的实体引用换成对应的实体内容.也就是说,一般的XML中的文本节点都是PCDATA,当这些文本中要包含一些XML特殊字符时,需要使用实体引用,当这些字符很少时,使用实体引用还不是很麻烦,而当特殊字符很多时,则需要使用CDATA,即不解析字符串. CDATA 区段开始于 "<![CDATA["，结束于 "]]>",CDATA段中可以包含除CDATA限定符之外的任何字符

元素声明

元素声明使用<!ELEMENT 元素名称 (元素内容)>或<!ELEMENT 元素名称 类别>的语法

```
<!ELEMENT root EMPTY> //EMPTY关键字表示元素是个空元素 <!ELEMENT root ANY> //ANY关键字表示元素中
   //下面这个声明表示root中可以有文本,也可以是空 <!ELEMENT root (#PCDATA)> //()表示一个分组,其中是
   //紧接着是a或b <!ELEMENT root (child,(a,b)+)> //*,?,+这些量词可作用于分组,这里表示root元素中
   //紧接着是a或b出现一次或多次 <![CDATA[ ABC-abc-ABC><&;&//这个不并被替换成&吧 ]]>
```

属性声明

属性声明使用<!ATTLIST 元素名称 属性名称 属性类型 默认值>的语法.示例如下:

```
<!ATTLIST input type CDATA "text">
```

上面的属性声明表示:元素input的type属性值是文本,默认值是text;

以下是属性类型表

类型	描述
CDATA	值为字符数据 (character data)
(en1|en2|..)	此值是枚举列表中的一个值
ID	值为唯一的 id
IDREF	值为另外一个元素的 id
IDREFS	值为其他 id 的列表
NMTOKEN	值为合法的 XML 名称
NMTOKENS	值为合法的 XML 名称的列表
ENTITY	值是一个实体
ENTITIES	值是一个实体列表
NOTATION	此值是符号的名称
xml:	值是一个预定义的 XML 值

默认值参数可以使用下列值

值	解释
值	属性的默认值.该属性可以出现,也可以不出现,当没有明确指定该属性时,属性值使用默认值
#REQUIRED	属性值是必需的
#IMPLIED	属性不是必需的,可以出现,可以不出现
#FIXED value	属性值是固定的.属性可有可无,但有的时候,其值必须是value

```
<!ATTLIST img src CDATA #REQUIRED> //img元素的src属性是必须的,值为字符串 <!ATTLIST script typ
```

实体声明

一般实体

实体是用于定义用于定义引用普通文本或特殊字符的快捷方式的变量。实体引用是对实体的引用。实体可在内部或外部进行声明。

一个内部实体是以<!ENTITY 实体名称 "实体的值">的形式声明的,一个外部私有实体是以<!ENTITY 实体名称 SYSTEM "URI/URL">格式声明 ,一个外部公共实体是以 <!ENTITY 实体名称 PUBLIC "公共实体名称" "URI/URL"> ,其中公共实体名称和DOCTYPE中的公共DTD名称格式是一样的. 一个实体引用是&实体名称;格式

```
//实体定义 <!ENTITY abc "ABCabcABC"> //内部实体 <!ENTITY abc SYSTEM "abc.ent"> //外部私有实体
   //实体引用 <abc>&abc;</abc>
```

参数实体

参数实体是只在DTD中使用的实体(并且参数实体只能在外部DTD中声明),它的声明语法与一般实体不同处在于其要在实体名称前加个百分号,而引用时则使用%实体名称;的形式

```
<!ENTITY % abc "root">
   <!ELEMENT %abc; (child)>//这句将声明元素root,具有一个子元素child
```

XML 命名空间

XML 命名空间可提供避免元素命名冲突的方法。由于 XML 中的元素名是预定义的，当两个不同的文档使用相同的元素名时，就会发生命名冲突。命名空间其实就是给这些标签名加个前缀!

```
<root>
       <svg:template />
       <xsl:template />
   </root>
```

现在,root下仍然是两个template元素,它们的节点名称仍然是template,但是它们的意义不一样了,因为它们使用了不同的前缀!但是XML命名空间前缀需要声明才可以使用,如果不声明,则被视为元素名称的一部分! XML 命名空间属性被放置于某个元素的开始标签之中，并使用以下的语法：xmlns:namespace-prefix="namespaceURI" . 当一个命名空间被定义在某个元素的开始标签中时，所有带有相同前缀的子元素都会与同一个命名空间相关联。 注意,用于标示命名空间的地址不会被解析器用于查找信息。其惟一的作用是赋予命名空间一个惟一的名称。不过，很多公司常常会作为指针来使用命名空间指向某个实存的网页，这个网页包含着有关命名空间的信息。

```
<root xmlns:svg="http://www.svg.org" xmlns:xsl="http://www.xsl.org">
       <svg:template />
       <xsl:template />
   </root>
```

这样,为了区分那些名称相同而含义不同的元素,必须在每个元素名前面加前缀.其实还可以在父级元素上声明默认命名空间,让所有没有前缀子元素的默认使用此命名空间.HTML的命名空间便是一个例子.

```
<html xmlns="http://www.w3.org/1999/xhtml">
   </html>
```

对于使用命名空间的XML文档，其DTD中对元素的声明也应该包含命名空间前缀(即应与文档中所书写的一致).另外,命名空间不但作用于元素,还作用于属性

javascript快速入门25--浏览器中的XML

打开XML

首先,直接从浏览器中打开XML文件,浏览器会对其进行格式良好性检查,如果不符合XML语法规范则显示出错,如果格式良好,再检查是否包含样式表(CSS或XSL),如果包含样式表,则用样式表格式化XML文档然后显示,如果没有,则显示经过格式化的XML源码(不同浏览器显示方式不一样).注意,浏览器只对XML进行格式良好性检查,而不对其进行有效性检查!如何在XML文档中引入样式表?示例:

```
<?xml version="1.0" standalone="no"?>
   <?xml-stylesheet type="text/css" href="test.css"?>
```

如果是使用XSL,只需将上面的type属性值改成text/xsl即可!

XMLHttpRequest对象的responseXML属性

XMLHttpRequest对象的responseXML属性用来获取从服务器端返回的XML文件,如果没有则为null.注意,只有在服务器环境下,请求一个以.xml为后缀的文件返回的才会是一个XML DOM,在本地请求一个XML文件,返回的仍是文本文件.如果要使用服务器脚本生成字符串,要返回XML,必须加上一些头信息

```
//JS Code
   var xhr = XHR();//XHR为之前所写的可以跨浏览器创建XMLHttpRequest对象的函数
   xhr.open("get","test.php",true);
   xhr.onreadystatechange = function () { if (xhr.readyState==4 && xhr.status == 200) {
           alert(xhr.responseXML.documentElement.nodeName);//可以使用DOM 2 Core的一些属性和
}
   };
   xhr.send(null); //PHP Code
   header("Content-Type:text/xml");//发送MIME信息
   echo  <<<XML <?xml version="1.0" encoding="gbk"?>
       <root />
   XML;
```

更简单的方法:直接加载XML文件

浏览器也提供了直接加载XML文件的方法,但也仅仅火狐与Windows平台上的IE支持而已.同样,两者之间的实现也有很大的差别的!

IE中的XML DOM

微软在JavaScript中引入了用于创建ActiveX对象的ActiveXObject类。ActiveXObject的构造函数只有一个参数——要进行实例化的ActiveX对象的字符串代号。例如，XML DOM对象的第一个版本称为Microsoft.XmlDom。所以，要创建这个对象的实例，使用以下代码：

```
var xmlDom = new ActiveXObject("Microsoft.XmlDom");
```

IE中的XML DOM支持的最新版本是5.0,IE中存在以下版本的XML DOM实现:

- Microsoft.XmlDom（最原始的）
- MSXML2.DOMDocument
- MSXML2.DOMDocument.3.0
- MSXML2.DOMDocument.4.0
- MSXML2.DOMDocument.5.0

自然地，只要可能，大家都会选择最新的XML DOM的版本，因为新版会提高速度，添加一些新的功能如验证等。但是，如果尝试创建不存在于客户端机器上的ActiveX对象，IE就会抛出错误并停止所有执行。所以，为确保使用了正确的XML DOM版本，也为避免任何其他错误，我们可以创建一个函数来测试每个XML DOM字符串，出现错误即捕获：

```
function XMLDOM() { var xmlDomVers = [ "Microsoft.XmlDom", "MSXML2.DOMDocument", "MSXML2
            } catch(e) {continue;}

        }
    }
```

接下来就是载入XML. 微软的XML DOM有两种载入XML的方法：loadXML()和load()。loadXML()方法可直接向XML DOM输入XML字符串.load()方法用于从服务器上载入XML文件。不过，load()方法只可以载入与包含JavaScript的页面存储于同一服务器上的文件，也就是说，不可以通过其他人的服务器载入XML文件。 load方法还有两种载入文件的模式：同步和异步。以同步模式载入文件时，JavaScript代码会等待文件完全载入后才继续执行代码；而以异步模式载入时，不会等待，可以使用事件处理函数来判断文件是否完全载入了。默认情况下，文件按照异步模式载入。要进行同步载入，只需设置async特性为false：

```
//Only For IE
    var xmlDom = XMLDOM();
    xmlDom.loadXML("<root />");
    alert(xmlDom.documentElement.nodeName); var xml = XMLDOM();
    xml.async = false;//同步载入
    xml.load("test.xml");
    alert(xml.documentElement.firstChild.nodeValue); //同步加载时需要使用readystatechange事件
    var xml2 = XMLDOM();
    xml2.async=true;//可以不指定,默认是异步载入的
    xml2.onreadystatechange = function () {//必须在调用load方法前分配此事件处理函数
        if (xml2.readyState==4) {//readyState的含义和XHR对象是一样的
            //注意这里没有使用this,因为IE下的ActiveXObject很特殊,使用this会出错
            alert(xml2.xml);//与innerHTML属性类似,IE中的xml属性返回XML字符串形式的源码
            //但注意,IE中的XMLDOM对象的xml属性是只读的
}
    };
    xml2.load("test.xml");
```

在尝试将XML载入到XML DOM对象中时，无论使用loadXML()方法还是load()方法，都有可能出现XML格式不正确的情况。为解决这个问题，微软的XML DOM的parseError的特性包含了关于解析XML代码时所遇到的问题的所有信息。

parseError特性实际上是包含以下特性的对象：

- errorCode——表示所发生的错误类型的数字代号（当没有错误时为0）
- filePos——错误发生在文件中的位置
- line——遇到错误的行号
- linepos——在遇到错误的那一行上的字符的位置
- reason——对错误的一个解释
- srcText——造成错误的代码
- url——造成错误的文件的URL（如果可用）

当直接对parseError自身取值，它会返回errorCode的值，也就是说可以这样进行检查

```
if (xmlDom.parseError===0) {
        alert("没有出错!");
    } else { var er = xmlDom.parseError;
        alert("XML解析出错!错误信息如下:\n\
        错误代号:"+er.errorCode+"\n\
        文件:"+er.filePos+"\n\
        行:"++er.line+"\n\
        字符:"+er.linepos+"\n\
        相关信息:"+er.reason+"\n\");
    }
```

Mozilla中的XML DOM

与Mozilla其他方面一样，它提供的XML DOM版本要比IE的更加标准。 Mozilla中的XML DOM实际上是它的JavaScript实现，也就是说它不仅与浏览器一起衍化，同时它能可靠地在Mozilla支持的所有平台上使用。因此与不能在Macintosh上使用XML DOM的IE不同，Mozilla的支持跨越了平台的界限。另外，Mozilla的XML DOM实现了支持DOM Level 2 的功能，而微软的，仅支持DOM Level 1。

DOM标准指出，document.implementation对象有个可用的createDocument()方法。Mozilla严格遵循了这个标准，可以这样创建XML DOM：

```
var xmlDom = document.implementation.createDocument("","",null);
```

createDocument()的三个参数分别是文档的命名空间URL，文档元素的标签名以及一个文档类型对象（总是为null，因为在Mozilla中还没有对文档类型对象的支持）。前面这行代码创建一个空的XML DOM。要创建包含一个文档元素的XML DOM，只需将标签名作为第二个参数：

```
var xmlDom = document.implementation.createDocument("","root",null);
```

这行代码创建了代表XML代码<root/>的XML DOM。如果在第一个参数中指定了命名空间URL，可进一步定义文档元素：

```
var xmlDom = document.implementation.createDocument("http://www.x-do.org","root",null);
```

这行代码创建了表示<root xmlns="http://www.x-do.org"/>的XML DOM

与微软的XML DOM不同，Mozilla只支持一个载入数据的方法：load()。Mozilla中的load()方法和IE中的load()工作方式一样。只要指定要载入的XML文件，以及同步还是异步（默认）载入。如果同步载入XML文件，代码基本上IE差不多：

```
var xmlDom = document.implementation.createDocument("","root",null);
   xmlDom.async = false;//同步载入
   xmlDom.load("test.xml");
```

与IE不同的是,同步载入时,并没有readychange事件,而只有load事件,并且只有加载完毕一种状态!

```
var xmlDom = document.implementation.createDocument("","root",null);
   xmlDom.async = true;//异步载入
   xmlDom.onload = function () {
       alert(this.documentElement.childNodes.item(0).tagName); //对于JS,访问节点列表既可以用
};
   xmlDom.load("test.xml");
```

另外,麻烦的是,Mozilla的XML DOM不支持loadXML()方法。要将XML字符串解析为DOM，必须使用DOMParser对象,使用其parseFromString方法传入XML字符串表现形式：

```
var xmlParser = new DOMParser(); var xmlDom = xmlParser.parseFromString("<root />","text
   //第二个参数text/xml也可以是application/xml,两者都用来解析XML
   //还可以是application/xhtml+xml,用来解析XHTML,只能用这三种MIME
```

与直接解析XML字符串相对应的获取XML字符串的方法,IE中XML DOM对象具有只读的xml属性,而Mozilla 则没有相对应的属性,但是，Mozilla提供了可以用于同样的目的的XMLSerializer对象：

```
var serializer= new XMLSerializer(); var xmlStr = serializer.serializeToString(xmlDom,"t
   //而text/xml也可为application/xml
```

对于XML 解析错误,Mozilla的实现方式非常麻烦,它不像IE那样提供一个对象来表示错误,而是将错误信息作为一个XML文档返回,要获取具体的错误信息,还必须用解析其中的字符串!

```
var xmlParser = new DOMParser(); var xmlDom = xmlParser.parseFromString("<root><child></
   alert(xmlDom.documentElement.nodeName);//将返回parsererror,因为文档解析出错时
   var serializer = new XMLSerializer(); var str = serializer.serializeToString(xmlDom,"
   alert(str);//将输出类似下面内容
   <parsererror xmlns="http://www.mozilla.org/newlayout/xml/parsererror.xml"> XML解析错误
   位置：file:///E:/XML/test.html
   行：1，列：16：<sourcetext>&lt;root&gt;&lt;child&gt;&lt;/root&gt;
   ---------------^</sourcetext></parsererror>
```

而从其中抽取出诸如行号这些错误信息,只好使用正则表达式了!但是由于浏览器的语言设置不同,使用正则表达式也是困难重重的! 可以看到,虽然Mozilla对XML DOM的实现更标准,但是使用起来是非常不方便的!

跨浏览器的XML DOM构造函数

说它跨浏览器,其实也仅仅是兼容Mozilla与Windows上的IE而已,而对于其它浏览器,则可以降级,考虑使用XHR的responseXML,虽然XHR对象该属性没有提供什么高级的方便的方法,但用于读取XML已经足够了!

```
if (document.implementation && document.implementation.createDocument) { //W3C
    var getXMLDOM=function () {//获取一个XMLDOM对象
        return document.implementation.createDocument("","",null);
    },
    loadXMLFile=function (xmlDom,url,callback) { if (xmlDom.async===true) {
            xmlDom.onload=function () { if (xmlDom.documentElement.nodeName=="parsererror
                } else {
                    callback.call(xmlDom);
                }
            };
        }
        xmlDom.load(url); return xmlDom;
    },
    loadXMLString=function (xmlDom,s) { var p = new DOMParser(); var newDom=p.parseFromSt
        } while (xmlDom.firstChild) {
            xmlDom.removeChild(xmlDom.firstChild);
        } for (var i=0,n;i<newDom.childNodes.length;i++) {
            n=xmlDom.importNode(newDom.childNodes[i],true); //importNode用于把其它文档中的节
            //true参数同时导入子节点
 xmlDom.appendChild(n);
        } return xmlDom;
    },
    getXML=function (xmlNode) { var s= new XMLSerializer(); return s.serializeToString(xm
    };
} else if (window.ActiveXObject) { //IE
    var getXMLDOM=function () { return new ActiveXObject("Microsoft.XmlDom");
    },loadXMLFile=function (xmlDom,url,callback) {
        xmlDom.onreadystatechange=function () { if (xmlDom.readyState===4) { if (xmlDom.p
                    callback.call(xmlDom);
                } else { throw new Error("XML Parse Error:"+xmlDom.parseError.reason);
                }
            }
        };
        xmlDom.load(url); return xmlDom;
    },loadXMLString=function (xmlDom,s) {
        xmlDom.loadXML(s); if (xmlDom.parseError.errorCode!==0) { throw new Error("XML Pa
        } return xmlDom;
    },
    getXML=function (xmlNode) { return xmlNode.xml;
    };
}
```

javascript快速入门26--XPath

XPath 简介

XPath 是一门在 XML 文档中查找信息的语言。XPath 可用来在 XML 文档中对元素和属性进行遍历。XPath 是 W3C XSLT 标准的主要元素，并且 XQuery 和 XPointer 同时被构建于 XPath 表达之上。因此，对 XPath 的理解是很多高级 XML 应用的基础。 其实对些我们并不陌生,最与XPath相似的便是CSS的选择器.在CSS中使用CSS选择符选择元素来应用样式,而在XSLT中则使用XPath,XPath与CSS选择器相比如强大的许多!下面是CSS选择符与XPath选择符一些对照:

```
//CSS选择符
   body p  //选择所有body下面的p元素
   body>p //选择body的子元素p
   * //选择所有的元素
   //与之对应的XPath选择符
   body//p
   body/p
   *
```

虽然现在还不能了解这些XPath表达的含意,但可以发现,它和CSS选择符十分相像!但XPath有更强大的地方,比如它可以定位到body元素下具体位置上的p或可以选择前N个p:

```
body/p[position()=4]  //这个XPath表达式将选取body子元素中第4个p元素,注意这里从1开始计数
   body/p[position()<3]  //将选取body子元素中前两个p元素
```

XPath 使用路径表达式来选取 XML 文档中的节点或者节点集。这些路径表达式和我们在常规的电脑文件系统中看到的表达式非常相似。另外,XPath 含有超过 100 个内建的函数。这些函数用于字符串值、数值，日期和时间比较、节点和 QName 处理、序列处理、逻辑值等等。

书写XPath

XPath使用路径表达式在 XML 文档中选取节点。节点是通过沿着路径(path)或者 步(step) 来选取的。如"/"表示文档节点,"."表示当前节点,而".."则表示当前节点的父节点.示例:

```
   {因为XPath表示达式中有斜杠,所以暂时用这个表示注释!
   / {选取文档节点,nodeType为9
   /root {选取文档根元素,类似于文件系统的路径(Unix),以/开头的路径都是绝对路径
   /root/child/.. {选取根节点root的子节点child的父节点(就是root)
```

下面是一些常用路径表达式

- nodeName 选取名称为nodeName的节点

- / 从根节点选取
- // 选择元素后代元素,必须在后面跟上nodeName
- . 选取当前节点
- .. 选取当前节点的父节点
- @ 选取属性节点(@是attribute的缩写)

```
<?xml version="1.0"?>
   <root>
       <child attr="attr" />
       <child>
           <a><desc /></a>
       </child>
   </root>
```

```
   {针对上面的XML文档的XPath结果,当前节点为document
   /root  {选取root
   root {选取root
   child {空,因为child不是document的子元素
   //child {选取两个child元素,//表示后代
   //@attr {选取attr属性节点
   /root/child//desc {返回child的后代元素desc
```

谓语（Predicates）

谓语用于在查找节点时提供更详尽的信息,谓语被嵌在方括号中。下面是一些带有谓语的XPath表达式:

```
/root/child[3] {选取root元素的第三个child子元素,注意,这和数组下标不一样,从1开始计数
   //child[@attr] {选取所有具有属性attr的child元素
   //child[@attr="val"]/desc {选取所有属性attr的值为val的child元素的子元素desc
   //child[desc] {选取所有的有desc子元素的child
   //child[position()>3]  {position()是XPath中的一个函数,表示节点的位置
   //child[@attr>12] {XPath表达式还可以进行数值比较,该表达式将选取attr属性值大于12的child元素
   //child[last()] {last()函数返回节点列表最后的位置,该表达式将选取最后一个child元素
```

通配符

XPath 通配符可用来选取未知的 XML 元素。

-
    - ,和CSS中的选择符一样,这将匹配任何元素节点
- @* ,匹配任何属性节点
- node() ,匹配任何类型的节点

```
/root/* {选取根元素下面的所有子元素
   /root/node() {选取根元素下面的所有节点,包括文本节点
   //* {选取文档中所有元素
   //child[@*] {选取所有具有属性的child元素
   //@*  {选取所有的属性节点
```

组合路径

与CSS中使用逗号组合使用多个选择符一样,XPath支持一种使用"|"来组合多个路径的语法!

```
/root | /root/child {选取根元素root与它下面的子元素child
   //child | //desc {选取所有的child元素与desc元素
```

XPath 运算符

下面列出了可用在 XPath 表达式中的运算符：

- | ,计算两个节点集
-
    - ,加法
-
    - ,减法
-
    - ,乘法
- div ,除法,因为/已经被作为路径符了,所以不能用来作为除法标识
- mod ,取余
- = ,等于
- != ,不等于
- < ,小于
- <= ,小于或等于
- > ,大于
- >= ,大于或等于
- or ,或
- and ,与

XPath 轴

轴可定义某个相对于当前节点的节点集。下面一可用的轴名称与对应的结果:

- ancestor 选取当前节点的所有先辈（父、祖父等）
- ancestor-or-self 选取当前节点的所有先辈（父、祖父等）以及当前节点本身
- attribute 选取当前节点的所有属性
- child 选取当前节点的所有子元素。
- descendant 选取当前节点的所有后代元素（子、孙等）。
- descendant-or-self 选取当前节点的所有后代元素（子、孙等）以及当前节点本身。
- following 选取文档中当前节点的结束标签之后的所有节点。
- namespace 选取当前节点的所有命名空间节点
- parent 选取当前节点的父节点。

- preceding 选取文档中当前节点的开始标签之前的所有节点。
- preceding-sibling 选取当前节点之前的所有同级节点。
- self 选取当前节点。

事实上,一个完整的XPath表达式由"/"与"步"构成的,而步又是由 "轴" 、 "节点测试"和"谓语"构成的.如下:

```
step/step/.....  {一个XPath表达式
{step的构成
轴名称::节点测试[谓语]
```

在一般的XPath表达式中,没有谓语即表达没有其它条件限制,而没有轴名称,则默认使用child.如"abc"与"child::abc"是等效的, 下面是一些与使用轴名称等效的简单XPath表达式:

- child::abc --------------------- abc(子元素abc)
- root/attribute::id ------------ root/@id(root的属性id)
- selft::node() ------------------ .(自身)
- parent::node() --------------- ..(父节点)
- child:: ------------------------ (子元素)
- child::text() ------------------ text()(子文本节点)
- descendant::tag ------------ .//tag (后代tag元素)

XPath还包含一套函数库,如position与last就是函数,一般的函数被用在谓语中,而在XSLT及XQuery中它们则得到了更广泛的使用.

浏览器中的XPath

IE浏览器对XPath的实现比较简单.一个XML DOM对象(及每个节点)都有selectSingleNode与selectNodes方法,传入XPath表达式,selectNodes返回匹配的节点列表,而selectSingleNode则只返回列表中第一个项目!

```
var xmlDom = getXMLDOM();//我们之前写的跨浏览器的XML DOM加载函数
   loadXMLFile(xmlDom,"text.xml"); var root = xmlDom.selectSingleNode("/*");//返回文档根元
   root = xmlDom.selectNodes("/*")[0];//同上
   var lastChild = xmlDom.selectSingleNode("/*/*[last()]");
```

Mozilla是根据DOM标准来实现对XPath的支持的。DOM Level 3附加标准DOM Level 3 XPath定义了用于在DOM中计算XPath表达式的接口。遗憾的是，这个标准要比微软直观的方式复杂得多。

虽然有好多与XPath相关的对象，最重要的两个是：XPathEvaluator和XPathResult。XPathEvaluator利用方法evaluate()计算XPath表达式。

evaluate()方法有五个参数：XPath表达式、上下文节点、命名空间解释程序和返回的结果的类型，同时，在XPathResult中存放结果（通常为null）。

命名空间解释程序，只有在XML代码用到了XML命名空间时才是必要的，所以通常留空，置为null。返回结果的类型，可以是以下十个常量值之一：

- XPathResult.ANY_TYPE——返回符合XPath表达式类型的数据
- XPathResult.ANY_UNORDERED_NODE_TYPE——返回匹配节点的节点集合，但顺序可能与文档中的节点的顺序不匹配
- XPathResult.BOOLEAN_TYPE——返回布尔值
- XPathResult.FIRST_ORDERED_NODE_TYPE——返回只包含一个节点的节点集合，且这个节点是在文档中第一个匹配的节点
- XPathResult.NUMBER_TYPE——返回数字值
- XPathResult.ORDERED_NODE_ITERATOR_TYPE——返回匹配节点的节点集合，顺序为节点在文档中出现的顺序。这是最常用到的结果类型
- XPathResult.ORDERED_NODE_SNAPSHOT_TYPE——返回节点集合快照，在文档外捕获节点，这样将来对文档的任何修改都不会影响这个节点列表。节点集合中的节点与它们出现在文档中的顺序一样
- XPathResult.STRING_TYPE——返回字符串值
- XPathResult.UNORDERED_NODE_ITERATOR_TYPE——返回匹配节点的节点集合，不过顺序可能不会按照节点在文档中出现的顺序排列
- XPathResult.UNORDERED_NODE_SNAPSHOT_TYPE——返回节点集合快照，在文档外捕获节点，这样将来对文档的任何修改都不会影响这个节点列表。节点集合中的节点和文档中原来的顺序不一定一样。

下面是使用ORDERED_NODE_ITERATOR_TYPE的例子:

```
var xmlDom = getXMLDOM();//我们之前写的跨浏览器的XML DOM加载函数
   loadXMLFile(xmlDom,"text.xml"); var evaluator = new XPathEvaluator(); var result =eva
       while(node=result.iterateNext()) {//这个列表必须使用iterateNext方法遍历
alert(node.tagName);
       }
   }
```

javascript快速入门27--XSLT基础

XSL 与 XSLT

XSL 指扩展样式表语言（EXtensible Stylesheet Language）。它主要被用来对XML文档进行格式化,与CSS不同,XSL不仅仅是样式表语言XSL主要包括三个部分:

- XSLT 一种用于转换 XML 文档的语言。 它可以将一个XML文件转换成另一种格式的XML文件或XHTML文件.
- XPath 一种用于在 XML 文档中导航,定位元素的语言。
- XSL-FO , 可扩展样式表语言格式化对象（Extensible Stylesheet Language Formatting Objects） ,用于格式化供输出的 XML 数据。XSL-FO 目前通常被称为 XSL (尽管这算是一种误解,但这样说是可以的,因为在格式化XML方面,XSL-FO起着和CSS一样的作用!)

XSLT 指 XSL 转换（XSL Transformations）。 它是 XSL 中最重要的部分。通过 XSLT，您可以向或者从输出文件添加或移除元素和属性。您也可重新排列元素，执行测试并决定隐藏或显示哪个元素，等等。描述转化过程的一种通常的说法是，XSLT 把 XML 源树转换为 XML 结果树。

书写 XSLT

XSLT 文件本身也是XML 文件,一般 以.xml .xsl .xslt几种文件后缀名保存.XSLT遵循XML的语法,文件开头一般都加有XML声明,XML声明之后是文档根元素stylesheet或transform(两者之一),并且使用version属性声明XSLT版本,目前版本是1.0,2.0还在草案中,XSLT的所有内置元素都从属于"http://www.w3.org/1999/XSL/Transform"命名空间,所以应该在文档根元素上声明一个xsl或xs的命名空间!

```
<?xml version="1.0" encoding="UTF-8"?>
   <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" />
```

上面创建了一个最基本的XSLT文件,将其应用于任何XML文档,在支持XSLT的浏览器打开该XML文档,会看到所有的文档显示了出来,而标签没有了!事实上,在浏览器中真正显示的是HTML,XSLT将XML转换成了HTML.我们可以更进一步指定转换成HTML的版本,比如转换成XHTML!

```
<?xml version="1.0" encoding="UTF-8"?>
   <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="html" encoding="utf-8" doctype-public="-//W3C//DTD XHTML 1.0 Tran
   </xsl:stylesheet>
```

output元素定义输出文档的格式。method属性可接受xml,html,text,name四种格式;version设置输出格式的 W3C 版本号（仅在 method="html" 或 method="xml" 时使用）; encoding设置输出中编码属性的值(对于HTML将会输出成charset的值); doctype-public规定 DTD 中要使用的公共标识符; doctype-system规定 DTD 中要使用的系统标识符; indent 规定在输出结果树时是否要增加空白,该值必须为 yes 或 no。

template 模版

可以用template来定义模版.template元素必须有match或name两个属性之一或两者都有,match属性用以并联XML中的元素,其值为一个XPath表达式,XPath表达式所选取的元素会被应用模版而进行转换. name属性为模版定义名称,用以在其它地方引用.一个template元素里面包含的是一些将被输出的HTML或XML标签.

```
<?xml version="1.0" encoding="UTF-8"?>
   <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="html" version="1.0" encoding="UTF-8" indent="yes"/>
       <xsl:template match="/bank/p/name">
           <strong>Name</strong>
       </xsl:template>
   </xsl:stylesheet>
```

对于使用一个没有任何模版的XSLT的XML文件,在浏览器中显示时只是简单的将其中的文本显示出来,应用了上面的XSLT之后,根元素bank下面子元素p的子元素name的值都将会显示成一个加粗的Name!而其它的则只是普通的文本.但这样并没有什么意义,我们还可以更进一步,将被XPath表达式"/bank/p/name"选取的元素的值显示出来-------value-of元素!value-of元素的select属性是必须的,属性值是一个XPath表达式,指定一个节点(如果是多节点,value-of中会获取第一个节点的值),然后将其里面的文本输出!将上面的模版替换成下面的,输出后就会将所有的name加粗!

注意**,**任何正文内容的输出都应该放在**template**元素里面**!**

```
<xsl:template match="/bank/p/name">
       <strong><xsl:value-of select="." /></strong>
   </xsl:template>
```

注意上面的value-of的select是使用的 "." ,而没有使用"/bank/p/name",因为"/bank/p/name"返回的是所有的name元素," . "表示模版当前应用的那个元素!

可以定义多个模版,如下:

```
<xsl:template match="/bank/p/name">
        <strong><xsl:value-of select="." />---</strong>
    </xsl:template>
    <xsl:template match="/bank/p/age">
        <em><xsl:value-of select="." />---</em>
    </xsl:template>
    <xsl:template match="/bank/p/money">
        <u><xsl:value-of select="." /></u><hr />
    </xsl:template>
```

当多个模版的**match**表达选取节点重叠时**,**后出现的模版的格式会覆盖先出现的**,**可以使用**template**元素的**priority**属性对模版的优先组进行编号**,**其值是一个数字**,**越大优先级越高**!**

这样,便给name,age,money这些元素都进行了格式化输出,但现在输出的HTML代码顺序仍然是按照在XML源文件中出现的顺序出现的.当需要对整个XML文档进行格式化输出的时候,可以将match属性设为"/"

```
<xsl:template match="/" />
```

使用上面的模版,将会使XML文档在浏览器中没有任何输出.可以在应用于根节点的模版中加上HTML标签,以输出完整的HTML文档!

```
<xsl:template match="/">
        <html>
        <head>
            <title>XSLT</title>
        </head>
        <body> 一个HTML页面 </body>
        </html>
    </xsl:template>
```

这样,即使定义了其它模版,它们的输出也不会出现在浏览器中,因为上面的模版覆盖了其它应用于子节点的模版的输出,要在其中包含其它模版的内容,可以使用XSLT的apply-templates元素来应用模版,该元素有两个属性,select属性值是一个XPath表达式,XPath表达式选取的节点及其子节点将被应用模版. 如果没有为这些节点定义模版,则直接输出节点的值.如果apply-templates元素不指定select属性,则将给当前节点(template元素的match属性所匹配的节点)的所有后代节点应用模版,如果没有定义模版,则直接输出所有节点的值.

下面的代码将直接输出所有节点的值

XSLT中有一个规定**:**如果一个节点没有任何可用的**template,**则将这个节点中所有文本节点的值返回**!**

```
<xsl:template match="/">
            <xsl:apply-templates />
    </xsl:template>
```

可以指定select属性,指明哪些节点将应用模版并输出在这个地方,这样就可以不以XML源文件中的顺序输出数据了!

```
<xsl:template match="/">
       <xsl:apply-templates select="/bank/p/money" />
           <hr />
       <xsl:apply-templates select="/bank/p/name" />
   </xsl:template>
```

还可以使用call-template调用指定的模版,call-template元素的name属性指定要调用模版的name

attribute 给元素添加属性

使用attribute元素,可以在转换时给元素动态添加属性!其语法很简单,下面是一个给img元素添加值为"test.jpg"的src属性的代码:

```
<img>
       <xsl:attribute name="source">test.jpg</xsl:attribute>
   </img>
```

for-each 节点遍历

XSLT中的for-each 元素允许您在 XSLT 中进行循环。该元素的select属性与其它元素的select属性一样,其值是一个XPath表达式,被XPath表达式选取的元素将被遍历!

```
<xsl:template match="/">
       <xsl:for-each select="/bank/p/name">
           <em><xsl:value-of select="." /></em><br />
       </xsl:for-each>
   </xsl:template>
```

上面的代码将遍历所有根元素bank的子元素p的name子元素并加以格式化后输出它的值. 注意,value-of元素的select的值"."表示选取当前节点,在for-each的内部,当前节点为for-each当前遍历到的节点!

sort 排序

sort 元素用于对结果进行排序。sort元素需要放在for-each元素内部.sort元素的select属性值为选取排序依据节点的XPath表达式,data-type属性有两个取值(text|number),指明是按字母顺序排序还是按数字大小排序! 另外,它还有个order属性,可以指定是按正顺还是倒序排序,取值为(ascending|descending),默认是ascending(正序)!

```
<xsl:for-each select="/bank/p">
       <xsl:sort select="./money" data-type="number" />
       <xsl:value-of select="./money" /><br />
   </xsl:for-each>
```

if 条件测试

在XSLT中还可以使用if元素进行条件判断,该元素的test属性值为一个条件测试XPath表达式,当值计算结果是真的时候才处理if元素中的内容!

```
<xsl:for-each select="/bank/p">
        <xsl:sort select="./money" data-type="number" order="descending" />
        <xsl:if test="position() &lt; 4 and age &gt;=18">
            <xsl:value-of select="./money" /><br />
        </xsl:if>
    </xsl:for-each>
```

上面的代码用以输出money排前三名的成年人. 注意,在if元素的test属性中,XPath表达中的一些特殊字符(如大于和小于)必须写成实体引用!

choose when...otherwise...... 多重条件测试

出于习惯,见到if语句可能会想到if...else语句,但XSLT中并没有if..else语句,取而代之的是即有if...else功能,又有switch..case功能的choose元素,choose元素有两个子元素when与otherwise,相当于 else if 与else ,或者,when相当于case语句,otherwise相当于default.when元素的test属性值同样是一个XPath表达式,当这个表达式返回真的时候,when的子元素才会显示!otherwise没有test属性,当所有的when元素的test都失败后,处理otherwise子元素!

```
<xsl:choose>
        <xsl:when test="name='PHPer'">PHPer就是PHP程序员的意思!</xsl:when>
        <xsl:when test="name='CJ'">好好Coding,天天向上!</xsl:when>
        <xsl:when test="name='DBD'">不懂!</xsl:when>
        <xsl:otherwise>其它人</xsl:otherwise>
    </xsl:choose>
```

浏览器中的 XSLT

只要有XML与XSLT解释引擎,就可以在任何地方使用任何语言利用XSLT将XML转换成HTML或其它文档,并且使用不同的语言并不会影响转换结果.也就是说,这种转换是与语言无关的,既可以在服务器端进行转换后,返回HTML页面,也可以客户端进行转换,它们的效果都是一样的.而且在客户端对XML文件进行转换,可以减轻服务器的负担.

在一个引入了XSLT文件的XML文件,浏览器会自动对其进行转换.但是,XML一般并不是在浏览器中显示,而是用来读取数据.当使用其它语言来手动转换时,需要将xml-stylesheet这样的PI去掉,这样,XML 文件可使用多个不同的 XSL 样式表来进行转换,增加了灵活性。

IE 中的XSLT

与IE支持XML DOM 一样,IE中XSLT相关API显得十分简单,同时IE对XSLT的支持也很有限!下面是在IE 中将一个XMLDOM使用XSLT转换成HTML的示例:

```
//载入XML数据文件
   var xml = new ActiveXObject("Microsoft.XMLDOM");
   xml.async = false;
   xml.load("test.xml"); //载入XSLT文件,XSLT也是作为XML文件载入的
   var xsl = new ActiveXObject("Microsoft.XMLDOM");
   xsl.async = false;
   xsl.load("test.xsl"); //直接在要转换的DOM上调用transformNode方法,传入XSLT DOM,返回字符串
   document.write(xml.transformNode(xsl));
```

Mozilla 中的XSLT

与Mozilla对XML DOM的支持一样,它对XSLT的支持更标准但更复杂!Mozilla使用一个XSLTProcessor对象来处理与XSLT有关的转换.

```
//载入XML数据
   var xml = document.implementation.createDocument("","",null);
   xml.async =false;
   xml.load("test.xml"); //载入XSLT
   var xsl = document.implementation.createDocument("","",null);
   xsl.async =false;
   xsl.load("test.xsl"); //创建XSLTProcessor
   var xslPro = new XSLTProcessor();
   xslPro.importStylesheet(xsl);//导入XSLT
   //使用transformToDocument将XML按XSLT进行转换,返回新文档的DOM
   var result = xslPro.transformToDocument(xml); //要将返回的DOM转换成字符串,还要使用XMLSeria
   var serializer =new XMLSerializer(); var html = serializer.serializeToString(result);
   document.write(html);
```

PHP 教程

第一章 如何加载运行已发布的PHP项目

一．安装AppServ2.5.10

参考安装文档

验证是否安装成功

Http://localhost:8090/index.php

http://localhost:8090/phpinfo.php

http://localhost:8090/phpmyadmin

username: root password:root

二．下载项目

到http://www.comsenz.com下载免费开源项目Discuz_7.2_FULL_SC_GBK.zip

a.把解压后的目录中的upload文件拷贝到D:\AppServ\www，并改名为discuz

b.登录到 http://localhost:8090/discuz/install，按操作步骤安装，设置管理员为

username: admin password: admin888

安装完成即可使用。

c.输入http://localhost:8090/discuz，选择默认版块。

看到下面的页面，说明安装成功。



三.到http://www.ecshop.com下载ECShop_V2.7.2_UTF8_Release0604.zip

a.把解压后的目录中的upload文件拷贝到D:\AppServ\www，并改名为ecshop.

b.登录到 http://localhost:8090/ecshop，按操作步骤安装，设置管理员为

username: admin password: admin888

安装完成即可使用。

c.输入http://localhost:8090/ecshop，看到如下页面，即安装成功。



四.生成一个自己的项目

打开zend studio 7.2,选择工作目录为：D:\AppServ\www

新建一个PHP project ，命名为Basic1

新建一个PHP File，命名为：demo1.php

输入如下内容：

```
<?php
    echo 'Hello World';
?>
```

打开浏览器，输入http://localhost:8090/Basic1/

可看到如下内容

```
Index of /Basic1

Name Last modified Size Description

Parent Directory

demo1.php 13-Dec-2011 00:33 31

orderform.php 13-Dec-2011 00:16 871

postorder.php 14-Dec-2011 22:29 1.1K

Apache/2.2.8 (Win32) PHP/5.2.6 Server at localhost Port 8090
```

点击demo1.php，可看到输出了Hello World.

五. 运行已有php文件

Orderform.php

```
<!DOCTYPE  html PUBLIC   "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/x
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>我的第一个PHP程序</title>
<style>
table {
    background:#ccc;
    width:200px;
    margin:20px auto;
}
table td {
    background:#fff;
}
</style>
</head>
<body>

<form method="post" action="postorder.php">
    <table>
       <tr><td>您的商品</td><td>价格</td><td>数量</td></tr>
       <tr><td>苹果</td><td>2.6元/斤</td><td><input type="text" size="5" name="apple" /></
       <tr><td>猪肉</td><td>13.2元/斤</td><td><input type="text" size="5" name="pig" /></t
       <tr><td>饼干</td><td>21元/盒</td><td><input type="text" size="5" name="biscuit" /><
       <tr><td colspan="3" align="center"><input type="submit" value="发送订单" /></td></t
    </table>
</form>

</body>
</html>
```

Postorder.php

```
<?php
    $apple=$HTTP_POST_VARS['apple'];
    $pig=$HTTP_POST_VARS['pig'];
    $biscuit=$HTTP_POST_VARS['biscuit'];
```

```
    $apple=$apple*2.6;
    $pig=$pig*13.2;
    $biscuit=$biscuit*21;

    $sum=$apple+$pig+$biscuit;
?>
<!DOCTYPE  html PUBLIC   "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/x
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>我的第一个PHP程序</title>
<style>
table {
    background:#ccc;
    width:200px;
    margin:20px auto;
}
table td {
    background:#fff;
}
</style>
</head>
<body>

<form method="post" action="postorder.php">
    <table>
       <tr><td>您的商品</td><td>价格</td><td>小记</td></tr>
       <tr><td>苹果</td><td>2.6元/斤</td><td><?echo $apple ?></td></tr>
       <tr><td>猪肉</td><td>13.2元/斤</td><td><?echo $pig ?></td></tr>
       <tr><td>饼干</td><td>21元/盒</td><td><?echo $biscuit ?></td></tr>
       <tr><td colspan="3" align="center">一共要支付<?echo $sum ?>元 [<a href="orderform.ph
    </table>
</form>

</body>
</html>
```

输入url：http://localhost:8090/Basic1/

看到如下页面

```
Index of /Basic1

Name Last modified Size Description

Parent Directory

demo1.php 13-Dec-2011 00:33 31

orderform.php 13-Dec-2011 00:16 871

postorder.php 14-Dec-2011 22:29 1.1K

Apache/2.2.8 (Win32) PHP/5.2.6 Server at localhost Port 8090
```

点击 orderform.php

您的商品	价格	数量
苹果	2.6元/斤	
猪肉	13.2元/斤	
饼干	21元/盒	
发送订单		

分别在数量一栏输入 2，点击发送订单，看到如下页面

您的商品	价格	小记
苹果	2.6元/斤	5.2
猪肉	13.2元/斤	26.4
饼干	21元/盒	42
一共要支付73.6元 [返回修改]		

第二章 PHP基础

一.在web页面嵌入PHP代码的几种风格

推荐使用标准风格或简短风格

```
<?php
    //标准风格
    echo 'Hello World!';
?>

<?
    //简短风格
    echo 'Hello World!';
?>

<script language="php">
    //script风格
    echo 'Hello World!';
</script>
```

二.代码注释的四种方式

```
<?php
    //单行注释

    /*
     * 多行注释
     */

    #shell风格注释

    /**
     * PHPdoc风格注释
     */
?>
```

三.向浏览器输出字符串的几种方法

```
<?php
    /*
     * echo函数的功能：向浏览器输出字符串
     * 函数返回值：void
     */
    echo 'echo function!';
    echo('<br/>');

    /*
     * echo函数的功能：向浏览器输出字符串
     * 函数返回值：int
     */
    print 'print function';
    echo('<br/>');
    echo print 'echo value of print function. ';
    echo('<br/>');

    /*
     * printf函数的功能：向浏览器输出字符串
     * 函数返回值：所打印字符串的长度
     */
    printf("a weekend have %d days",7);
    echo('<br/>');
    echo printf("a weekend have %d days",7);
    echo('<br/>');

    /*
     * sprintf函数的功能：把字符串保存到内存中
     * 函数返回值：保存的字符串本身
     */
    sprintf('sprintf function');
    echo('<br/>');
    echo sprintf('sprintf function');
    echo('<br/>');
?>
```

输出结果：

```
echo function test!
print function test.
print function test. 1
a weekend have 7 days
a weekend have 7 days. 23

sprintf function test
```

常用类型指示符

类型	描述
%b	整数，显示为二进制
%c	整数，显示为ASCII字符
%d	整数，显示为有符号十进制数
%f	浮点数，显示为浮点数
%o	整数，显示为八进制数
%s	字符串，显示为字符串
%u	整数，显示为无符号十进制数
%x	整数，显示为小写的十六进制数
%X	整数，显示为大写的十六进制数

四.标识符与变量

1.标识符的基本规则：

1） 标识符可以是任意长度，而且可以由任何字母、数字、下划线组成。

2） 标识符不能以数字开始。

3） 在PHP中，标识符是区分大小写的。

4） 一个变量名称可以与一个函数名称相同。

2.变量赋值：

```
<?php

    $sum = 0;

    $total = 1.22;

    $sum = $total;

    echo $sum;  //1.22

?>
```

3.变量的数据类型:

基本数据类型

类型	名称
Integer	整数
Float	单精度浮点数
Double	又精度浮点数
String	字符串
Boolean	布尔
Array	数组
Object	对象

4.类型强度

PHP是动态语言，是一种非常弱的类型语言，在程序运行时，可以动态的改变变量的类型。

5.类型转换：

隐式类型转换：

```
<?php
$sum = 0;
$total = 1.22;
$sum = $total;
echo gettype ( $sum );//double
?>
```

显式类型转换：

```
<?php
$sum = 100;
$total = ( string ) $sum;
echo gettype ( $sum );//string
?>
```

使用settype()函数进行类型转换，返回值1表示成功，空表示失败。

```
<?php
$sum = 58;
echo settype ( $sum, "float" );
echo $sum; //58
echo gettype ( $sum ); //double
?>
```

6.检测变量的函数：

函数	功能	返回值
Gettype()	获取变量的类型	基本数据类型中的其中一种
Settype()	设置变量的类型	Bool(1:true 0:false(or ”))
Isset()	用来判断一个变量是否存在	Bool
Unset()	释放给定的变量	Void
Empty()	检测一个变量的值是否为空	Bool
is_int() is_integer()	检测变量是否是整数	Bool
Is_string()	检测变量是否是字符串	bool
Is_numeric	检测变量是否为数字或数字字符串	bool
Is_null	检测变量是否为 NULL	bool
Intval()	获取变量的整数值	int

Isset()的基本使用

```
<?php
$a = 10;
echo isset ( $a );//1
?>

<?php
echo isset ( $b );//''
?>
```

Usset()的基本使用

```
<?php
$a = 10;

unset($a);

echo isset ( $a );//''
?>
```

Empty()的基本使用

```
<?php
$a= 5;
$b =1;
$c = 0;
$d = "";
$e = array();
$f = null;
$g = "0";
$h = false;
echo empty($a);//''(false)
echo '<br/>';
echo empty($b);//''(false)
echo '<br/>';
echo empty($c);//1(true)
echo '<br/>';
echo empty($d);//1(true)
echo '<br/>';
echo empty($e);//1(true)
echo '<br/>';
echo empty($f);//1(true)
echo '<br/>';
echo empty($g);//1(true)
echo '<br/>';
echo empty($h);//1(true)
echo '<br/>';
echo empty($f);//1(true)
?>
```

is_int()的基本使用。类似的函数有:is_float()、is_double()、is_string()、is_bool()、is_array()、is_null()、is_long()、is_object()、is_resource()、is_numeric()、is_real()等。

```
<?php
$a = 11;
$b = 1.23;
$c = 3.1415926;
$d = "hello";
$e = false;
$f = array();
$g = null;

echo is_int($a);//1
echo '<br/>';
echo is_float($b);//1
echo '<br/>';
echo is_double($c);//1
echo '<br/>';
echo is_string($d);//1
echo '<br/>';
echo is_bool($e);//1
echo '<br/>';
echo is_array($f);//1
echo '<br/>';
echo is_null($g);//1
echo '<br/>';
echo is_numeric($a);//1
?>
```

Intval()函数的基本使用。类似的函数为：floatval()、strval()

```
<?php
$a = 22.23;
echo gettype($a);//double
echo '<br/>';
$b = intval($a);//类型转换后不改变$a原来的类型
echo gettype($a);//double
echo '<br/>';
?>

<?php
$a = 22.23;
echo gettype($a);//double
echo '<br/>';
settype($a,"integer");//类型转换后会改变$aa原来的类型
echo gettype($a);//integer
echo '<br/>';
?>
```

7.变量的作用域

超级全局变量

变量名	作用
$GLOBALS	所有全局变量数组
$_SERVER	服务器环境变量数组
$_GET	通过GET方式传递给该脚本的变量数组
$_POST	通过POST方式传递给该脚本的变量数组
$_COOKIE	COOKIE变量数组
$_FILES	与文件上传相关的变量数组
$_ENV	环境变量数组
$_REQUEST	所用用户输入的变量数组
$_SESSION	会话变量数组

8.常量

一旦被定义之后，就不能再次更改。

```
<?php
define("TOTAL",100);
echo TOTAL;//100
echo '<br/>';

define("TOTAL",200);
echo TOTAL;//100
?>
```

查看PHP预定义的常量的方法

```
<?php
phpinfo();
?>
```

引用PHP预定义常量的方法

```
<?php
echo $_SERVER["SERVER_NAME"];//localhost
echo '<br/>';
echo $_SERVER["SERVER_PORT"];//8090
echo '<br/>';
echo $_SERVER["DOCUMENT_ROOT"];//D:/AppServ/www
echo '<br/>';
?>
```

五.访问表单变量

常见的三种方式

```
<?php
echo $username;//简短风格，容易与变量名混淆，不推荐使用。
echo '<br/>';
echo $_POST['username'];//中等风格，4.1.0版后支持，推荐
echo '<br/>';
echo $HTTP_POST_VARS['username'];//冗长风格，已过时，将来可能会被剔除
?>
```

Posttest.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>获取表单数据的方式</title>
</head>
<body>
<form method="POST" action="demo10.php">
    username:<input type="text" name="username" />
    <input type="submit" value="submit" />
</form>
</body>
</html>
```

六.字符串连接用.

```
<?php
echo "the student name is ：".$_POST['username'];
echo "<br/>";
echo "welcome to "."school";
?>
```

第三章 操作符与控制结构

一．字符串插入

双引号与单引号的区别：

1.双引号的使用：

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<?php
//双引号可以解析变量和转义字符
$username = "jack";
echo "his name is $username!";
echo "<br/>";

$username = "小东";
//如果是英文的感叹号会正常解析变量
echo "他的名字是$username!";//他的名字是小东!
echo "<br/>";
//如果是中文的感叹号则会解析不出来
echo "他的名字是$username！";//他的名字是
echo "<br/>";

//转义字符在这里虽然被解析出来了，但是\n是在源代码里换行
//浏览器显示只是一个字符的位置
echo "他的名字是$username,\n他今年20岁了";//他的名字是小东, 他今年20岁了
echo "<br/>";

//为了避免出现错误，推荐使用字符串连接的方式
echo "他的名字是".$username.",他今年20岁了";//他的名字是小东,他今年20岁了
?>
```

2.单引号的使用：

```
<?php
//单引号只是输出字符串字面值，
//不会解析变量和转义字符。
//也不会进行语法加亮提示
$username = 'anllin';
echo 'his name is $username,\n his age is 20.';
//output
//his name is $username,\n his age is 20.
?>
```

部分常用的转义字符

转义序列	描述
\n	换行符
\r	回车
\t	水平制表图
\	反斜杠
$	美元符
\”	双引号

二 · 操作符

```
<?php
    //算术操作符
    $a = 5;
    $b = 3;
    echo $a + $b;
    echo '<br/>';
    echo $a - $b;
    echo '<br/>';
    echo $a * $b;
    echo '<br/>';
    echo $a / $b;
    echo '<br/>';
    echo $a % $b;
?>
```

```
8
2
15
1.66666666667
2
```

```
<?php
    //复合赋值操作符
    $a = 5;
    $b = 3;
    echo $a += $b;
    echo '<br/>';
    echo $a -= $b;
    echo '<br/>';
    echo $a *= $b;
    echo '<br/>';
    echo $a /= $b;
    echo '<br/>';
    echo $a %= $b;
    echo '<br/>';
    echo $a .= $b;
?>
```

```
8
5
15
5
2
23
```

```
<?php
    //递增递减运算符
    $a = 5;
    echo ++$a;
    echo '<br/>';
    echo $a++;
    echo '<br/>';
    echo --$a;
    echo '<br/>';
```

```
    echo $a--;

?>

6
6
6
6
```

```
<?php

$a = 5;

$b = 3;

$c = 5;

$d = '5';


echo $a == $c;

echo '<br/>';

echo $a === $c;

echo '<br/>';

echo $a == $d;

echo '<br/>';

echo $a != $b;

echo '<br/>';

echo $a !== $d;

echo '<br/>';

echo $a != $b;

echo '<br/>';

echo $a > $b;

echo '<br/>';

echo $b < $c;

echo '<br/>';

echo $a >= $c;

echo '<br/>';

echo $a <= $c;

?>

1
1
1
```

```
1
1
1
1
1
1
1




<?php


$a = false;

echo ! $a;

echo '<br/>';


$b = 5;

$c = 3;

echo $b > 0 && $c > 0;

echo '<br/>';

echo $b > 0 and $c > 0;

echo '<br/>';


echo $b != 0 || $c != 0;

echo '<br/>';

echo $b != 0 or $c != 0;

echo '<br/>';


?>

1
1
1
1
1
```

运算符”and”和”or”比&&和||的优先级要低

三元操作符

```
<?php
$a = 100;
echo $a > 60 ? 'success':'fail';
?>
success
```

错误抑制操作符

```
<?php
echo @(100/0);
?>
```

数组操作符			
操作符	使用方法	使用方法	说明
+	联合	!$b	返回一个包含了$a 和$b 中所有元素的数组
==	等价	$a&&$b	如果$a 和$b 具有相同的元素，返回 true
===	恒等	$a\|\|$b	如果$a 和$b 具有相同的元素以及相同的顺序，返回 true
!=	非等价	$a and $b	如果$a 和$b 不是等价的，返回 true
<>	非等价		如果$a 和$b 不是等价的，返回 true
!==	非恒等	$a or $b	如果$a 和$b 不是恒等的，返回 true

操作符优先级

<table>
<tr><th>结合性</th><th>操作符</th></tr>
<tr><td>左</td><td>,</td></tr>
<tr><td>左</td><td>Or</td></tr>
<tr><td>左</td><td>Xor</td></tr>
<tr><td>左</td><td>And</td></tr>
<tr><td>右</td><td>Print</td></tr>
<tr><td>左</td><td>= += -= *= /= .= %= &= |= ^= ~= <<= >>=</td></tr>
<tr><td>左</td><td>? :</td></tr>
<tr><td>左</td><td>||</td></tr>
<tr><td>左</td><td>&&</td></tr>
<tr><td>左</td><td>|</td></tr>
<tr><td>左</td><td>^</td></tr>
<tr><td>左</td><td>&</td></tr>
<tr><td>不相关</td><td>== != === !==</td></tr>
<tr><td>不相关</td><td><<= >>=</td></tr>
<tr><td>左</td><td><< >></td></tr>
<tr><td>左</td><td>+ - .</td></tr>
<tr><td>左</td><td>* / %</td></tr>
<tr><td>右</td><td>! ~ ++ --
(int)(double)(string)(array)(object) @</td></tr>
<tr><td>右</td><td>[]</td></tr>
<tr><td>不相关</td><td>New</td></tr>
<tr><td>不相关</td><td>()</td></tr>
</table>

三．控制结构

If条件判断语句

```
<meta http-equiv="content-type" content="text/html;charset=utf-8"/>
<?php


$a = 10;
if ($a > 0)
{
    echo '整数大于零';
}
echo '<br/>';


if ($a > 0)
{
    echo '整数大于零';
}
else if($a < 0)
{
    echo '整数小于零';
}
else
{
    echo '整数等于零';
}


?>
```

Switch语句

```
<meta http-equiv="content-type" content="text/html;charset=utf-8"/>
<?php
$role = 'admin';

switch ($role)
{
    case 'admin' :
       echo '管理员';
       break;
    case 'user' :
       echo '普通用户';
       break;
    case 'guest' :
       echo '游客';
       break;
    default :
       echo '游客';
       break;
}
?>
```

While循环语句

```
<?php
$a = 10;
while ( $a > 0 )
{
    echo $a --;
    echo '<br>';
}
?>
```

Do while 循环语句

```
<?php
$a = 10;
do
{
    echo $a --;
    echo '<br/>';
}
while ( $a > 0 )
?>
```

For循环语句

```
<?php
for($a = 0; $a < 10; $a++)
{
    echo $a;
    echo '<br/>';
}
?>
```

Break语句

```
<meta http-equiv="content-type" content="text/html;charset=utf-8"/>
<?php
for($a = 0; $a < 10; $a++)
{
    echo $a;
    echo '<br/>';
    if($a ==5)
    {
       break;//终止循环，但执行循环后面的语句
    }
}

echo '循环结束';
?>
```

Exit语句

```
<?php
for($a = 0; $a < 10; $a++)
{
    echo $a;
    echo '<br/>';
    if($a ==5)
    {
        exit;//直接退出，循环后面的语句不执行
    }
}

echo '循环结束';
?>
```

Continue语句

```
<?php
for($a = 0; $a < 10; $a++)
{
    echo $a;
    echo '<br/>';
    if($a ==5)
    {
        continue;//结束本次循环，继续下次循环，循环后面的语句依然执行
    }
}

echo '循环结束';
?>
```

第四章 数学运算

一.数值数据类型

数字或数值数据在PHP中一般就两种double和int。

PHP是一种松散类型的脚本语言，要注意类型转换的方式。

```
<?php
$a = '5';
//数字的字符串也是数字，参与数学运算当数字处理
echo is_numeric ( $a ); //1
echo '<br/>';
echo 7 + $a; //12
echo '<br/>';
echo '7' + $a; //12
echo '<br/>';
//用.连接后就按字符串处理
echo '7' . $a; //75
?>
```

二.随机数

Rand()函数是libc中定义的一个随机函数的简单包装器。

Mt_rand()函数是一个很好的代替实现。

```
<?php

$a = rand(0,10);

echo $a;

echo '<br/>';

echo getrandmax();

echo '<br/>';


$b = mt_rand(0,10);

echo $b;

echo '<br/>';

echo mt_getrandmax();

echo '<br/>';

?>
```

output

```
1
32767
6
2147483647
```

三.格式化数据

```
<?php

$a = 12345.6789;

//用于设置保留多少位小数点

echo number_format($a,2);

echo '<br/>';

//也可以改变默认小数点的符号表示和千分位的表示符号

echo number_format($a,2,'#','*')

?>
```

Output

```
12,345.68
12*345#68
```

四・数学函数

函数	功能
Abs()	取绝对值
Floor()	舍去法取整
Ceil()	进一法取整
Round()	四舍五入
Min()	求最小值或数组中最小值
Max()	求最大值或数组中最大值

```
<?php
$a = -123456.789;
$b = array (1, 2, 3, 4 );
echo abs ( $a );
echo '<br/>';
echo floor ( $a );
echo '<br>';
echo ceil ( $a );
echo '<br>';
echo round ( $a );
echo '<br>';
echo min ( $b );
echo '<br>';
echo max ( $b );
?>
```

output

```
123456.789
-123457
-123456
-123457
1
4
```

第五章 数组

一·什么是数组

数组是一组有某种共同特性的元素，包括相似性和类型。

每个元素由一个特殊的标识符来区分，称之为key，而每个key都有一个value

1.创建数组的两种方式：

1.1 用array()函数

```
<?php
$usernames = array ('Alerk', 'Mary', 'Lucy', 'Bob', 'Jack', 'John', 'Mark' );
foreach ( $usernames as $name )
{
    echo $name . '<br/>';
}
?>
```

output

```
Alerk
Mary
Lucy
Bob
Jack
John
Mark
```

1.2 用range()函数

```
<?php
$numbers = range ( 0, 10 );
foreach ( $numbers as $num )
{
    echo $num . '<br/>';
}


$letters= range ( 'a', 'z' );
foreach ( $lettersas $letter )
{
    echo $letter . '<br/>';
}
?>
```

output

```
0
1
2
3
4
5
6
7
8
9
10
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
```

2.循环访问数组元素的两种方式:

2.1 for循环

```
<?php
//range的第三个参数表示步长
$numbers = range(1,10,2);
for($i = 0;$i<count($numbers); $i ++)
{
    echo $numbers[$i].'<br/>';
}
?>
```

output

```
1
3
5
7
9
```

2.2 foreach循环

```
<?php
$letters = range('a','h',2);
foreach($letters as $letter)
{
    echo $letter.'<br/>';
}
?>
```

output

```
a
c
e
g
```

Foreach还可以用来输出数组的下标和对应的值

```
<?php
$letters = range('a','g',2);
foreach($letters as $key => $value)
{
    echo $key.'---'.$value.'<br/>';
}
?>
```

output

```
0---a
1---c
2---e
3---g
```

3.is_array()函数，用于变量判断是否为一个数组

```
<?php
$numbers = range(1,10,2);
if(is_array($numbers))
{
    foreach($numbers as $num)
    {
        echo $num.'<br/>';
    }
}
else
{
    echo $numbers;
}
?>
```

4.print_r函数，打印关于变量的易于理解的信息

```
<?php
$usernames = array ('Jackie', 'Mary', 'Lucy', 'Bob', 'Mark', 'John' );
print_r ( $usernames );
?>
```

output

```
Array ( [0] => Jackie [1] => Mary [2] => Lucy [3] => Bob [4] => Mark [5] => John )
```

源代码里可以看到显示为：

```
Array
(
    [0] => Jackie
    [1] => Mary
    [2] => Lucy
    [3] => Bob
    [4] => Mark
    [5] => John
)
```

二·自定义键数组

1.如果不想创建默认下标为零的数组，可以用如下方法，创建键为字符串的数组

```
<?php
//初始化数组
$userages = array('Jack'=> 23,'Lucy'=>25,'Mark'=>28);

//访问数组各元素
echo $userages['Jack'].'<br/>';
echo $userages['Lucy'].'<br/>';
echo $userages['Mark'].'<br/>';

?>
```

2.往自定义键数组里追加元素

```
<?php
//初始化数组
$ages = array('Jack'=>23);
//追加元素
$ages['Lucy']=25;
$ages['Mark']=28;

foreach($ages as $key => $value)
{
    echo $key.'----'.$value.'<br/>';
}
?>
```

3.直接添加元素，无需创建数组。

```
<?php
//不创建数组直接添加
$ages['Jack']=23;
$ages['Lucy']=25;
$ages['Mark']=28;
foreach($ages as $key => $value)
{
    echo $key.'----'.$value.'<br/>';
}
?>
```

4.循环打印数组foreach的使用

```
<?php
$ages['Jack']=23;
$ages['Lucy']=25;
$ages['Mark']=28;

foreach($ages as $key => $value)
{
    echo $key.'=>'.$value.'<br/>';
}

?>
```

5. each() -- 返回数组中当前的键／值对并将数组指针向前移动一步

```
<?php
$ages['Jack']=23;
$ages['Lucy']=25;
$ages['Mark']=28;

$a = each($ages);
print_r($a);
echo '<br/>';
$a = each($ages);
print_r($a);
echo '<br/>';
$a = each($ages);
print_r($a);
?>
```

用each()函数做循环打印

```
<?php
$ages['Jack']=23;
$ages['Lucy']=25;
$ages['Mark']=28;

while(!! $element = each($ages))
{
    print_r($element);
    echo '<br>';
}
?>
```

另一种打印方式

```
<?php
$ages['Jack']=23;
$ages['Lucy']=25;
$ages['Mark']=28;

while(!! $element = each($ages))
{
    echo $element['key'].'=>'.$element['value'];
    echo '<br>';
}
?>
```

6.list()函数的使用--把数组中的值赋给一些变量

```
<?php
$ages['Jack']=23;
$ages['Lucy']=25;
$ages['Mark']=28;

list($name,$age)= each($ages);
echo $name.'=>'.$age;
?>
```

用list循环打印结果

```
<?php
$ages['Jack']=23;
$ages['Lucy']=25;
$ages['Mark']=28;

while(!!list($name,$age)= each($ages))
{
    echo $name.'=>'.$age.'<br>';
}
?>
```

output

```
Jack=>23
Lucy=>25
Mark=>28
```

7.reset()函数的使用--将数组的内部指针指向第一个单元

```
<?php
$ages['Jack']=23;
$ages['Lucy']=25;
$ages['Mark']=28;

each($ages);
each($ages);
list($name,$age)= each($ages);
echo $name.'=>'.$age.'<br>';
//把数组重新设定到数组开始处
reset($ages);
list($name,$age)= each($ages);
echo $name.'=>'.$age.'<br>';
?>
```

Output

```
Mark=>28
Jack=>23
```

8. array_unique() -- 移除数组中重复的值

```
<?php
$nums = array(1,2,3,4,5,6,5,4,3,2,1,1,2,3,4,5,6);
//返回一个不包含重复值的数组
$result = array_unique($nums);
print_r($result);
?>
```

Output

```
Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5 [5] => 6 )
```

9. array_flip ()-- 交换数组中的键和值

```
<?php
$userages = array('Jack'=> 23,'Lucy'=>25,'Mark'=>28);


$ages = array_flip($userages);


print_r($ages);
?>
```

output

```
Array ( [23] => Jack [25] => Lucy [28] => Mark )
```

三·数组里的数组

数组里不一定就是一个关键字和值的列表，数组里也可以放入数组

```
<?php
$produces = array(
    array('apple',6,28.8),
    array('pear',3,15.6),
    array('banana',10,4.6)
);


echo $produces[0][0].'|'.$produces[0][1].'|'.$produces[0][2].'<br>';
echo $produces[1][0].'|'.$produces[1][1].'|'.$produces[1][2].'<br>';
echo $produces[2][0].'|'.$produces[2][1].'|'.$produces[2][2].'<br>';
?>
```

output

```
apple|6|28.8
pear|3|15.6
banana|10|4.6
```

用for循环打印数组中的数组

```
<?php
$produces = array (
    array ('apple', 6, 28.8 ),
    array ('pear', 3, 15.6 ),
    array ('banana', 10, 4.6 )
);

for($i = 0; $i < count ( $produces ); $i ++)
{
    for($j = 0; $j < count ( $produces [$i] ); $j ++)
    {
       echo '|' . $produces[$i][$j];
    }
    echo '<br>';
}
?>
```

output

```
|apple|6|28.8
|pear|3|15.6
|banana|10|4.6
```

二维数组

```
<?php
$produces = array (
    array ('name' => 'apple', 'amount' => 6, 'price' => 28.8 ),
    array ('name' => 'pear', 'amount' => 3, 'price' => 15.6 ),
    array ('name' => 'banana', 'amount' => 10, 'price' => 4.6 )
);

while(!!List($key,$value)=each($produces))
{
    while(!!list($key2,$value2)=each($value))
    {
       echo '|'.$key2.'=>'.$value2;
    }
    echo '<br>';
}
?>
```

output

```
|name=>apple|amount=>6|price=>28.8
|name=>pear|amount=>3|price=>15.6
|name=>banana|amount=>10|price=>4.6
```

用foreach来打印则更容易（推荐）

```
<?php
$produces = array (
    array ('name' => 'apple', 'amount' => 6, 'price' => 28.8 ),
    array ('name' => 'pear', 'amount' => 3, 'price' => 15.6 ),
    array ('name' => 'banana', 'amount' => 10, 'price' => 4.6 )
);

foreach($produces as $key1 => $value1)
{
    foreach($value1 as $key2 => $value2)
    {
       echo '|'.$key2.'=>'.$value2;
    }
    echo '<br>';
}
?>
```

output

```
|name=>apple|amount=>6|price=>28.8
|name=>pear|amount=>3|price=>15.6
|name=>banana|amount=>10|price=>4.6
```

四・数组的排序

1.Sort()函数对英文的排序

```
<meta http-equiv="content-type" content="text/html;charset=utf-8" />
<?php
$fruits = array('lemo','banana','apple','pear');
echo '原始的数组：';
print_r($fruits);
echo '<br/>';
sort($fruits);
echo '排序后的数组：';
print_r($fruits);
?>
```

output

```
原始的数组：Array ( [0] => lemo [1] => banana [2] => apple [3] => pear )
排序后的数组：Array ( [0] => apple [1] => banana [2] => lemo [3] => pear )
```

2.Sort()函数对中文的排序

```
<meta http-equiv="content-type" content="text/html;charset=utf-8" />

<?php

$fruits = array('柠檬','香蕉','苹果','梨子');

echo '原始的数组：';

print_r($fruits);

echo '<br/>';

sort($fruits);

echo '排序后的数组：';

print_r($fruits);

?>
```

Output:

```
原始的数组：Array ( [0] => 柠檬 [1] => 香蕉 [2] => 苹果 [3] => 梨子 )
排序后的数组：Array ( [0] => 柠檬 [1] => 梨子 [2] => 苹果 [3] => 香蕉 )
```

3. asort -- 对数组进行排序并保持索引关系

```
<meta http-equiv="content-type" content="text/html;charset=utf-8" />

<?php

$fruits = array('a'=>'柠檬','b'=>'香蕉','c'=>'苹果','d'=>'梨子');

echo '原始的数组：';

print_r($fruits);

echo '<br/>';

asort($fruits);

echo '排序后的数组：';

print_r($fruits);

?>
```

output

```
原始的数组：Array ( [a] => 柠檬 [b] => 香蕉 [c] => 苹果 [d] => 梨子 )
排序后的数组：Array ( [a] => 柠檬 [d] => 梨子 [c] => 苹果 [b] => 香蕉 )
```

4. ksort -- 对数组按照键名排序

```
<meta http-equiv="content-type" content="text/html;charset=utf-8" />

<?php

$fruits = array('b'=>'柠檬','a'=>'香蕉','d'=>'苹果','c'=>'梨子');

echo '原始的数组：';

print_r($fruits);

echo '<br/>';

ksort($fruits);

echo '排序后的数组：';

print_r($fruits);

?>
```

output

```
原始的数组：Array ( [b] => 柠檬 [a] => 香蕉 [d] => 苹果 [c] => 梨子 )
排序后的数组：Array ( [a] => 香蕉 [b] => 柠檬 [c] => 梨子 [d] => 苹果 )
```

5. rsort -- 对数组逆向排序

```
<meta http-equiv="content-type" content="text/html;charset=utf-8" />

<?php

$fruits = array('柠檬','香蕉','苹果','梨子');

echo '原始的数组：';

print_r($fruits);

echo '<br/>';

rsort($fruits);

echo '排序后的数组：';

print_r($fruits);

?>
```

output

```
原始的数组：Array ( [0] => 柠檬 [1] => 香蕉 [2] => 苹果 [3] => 梨子 )
排序后的数组：Array ( [0] => 香蕉 [1] => 苹果 [2] => 梨子 [3] => 柠檬 )
```

6. arsort -- 对数组进行逆向排序并保持索引关系

```
<meta http-equiv="content-type" content="text/html;charset=utf-8" />

<?php

$fruits = array('a'=>'柠檬','b'=>'香蕉','c'=>'苹果','d'=>'梨子');

echo '原始的数组：';

print_r($fruits);

echo '<br/>';

arsort($fruits);

echo '排序后的数组：';

print_r($fruits);

?>
```

output

```
原始的数组：Array ( [a] => 柠檬 [b] => 香蕉 [c] => 苹果 [d] => 梨子 )
排序后的数组：Array ( [b] => 香蕉 [c] => 苹果 [d] => 梨子 [a] => 柠檬 )
```

7. krsort -- 对数组按照键名逆向排序

```
<meta http-equiv="content-type" content="text/html;charset=utf-8" />

<?php

$fruits = array('a'=>'柠檬','b'=>'香蕉','c'=>'苹果','d'=>'梨子');

echo '原始的数组：';

print_r($fruits);

echo '<br/>';

krsort($fruits);

echo '排序后的数组：';

print_r($fruits);

?>
```

output

```
原始的数组：Array ( [a] => 柠檬 [b] => 香蕉 [c] => 苹果 [d] => 梨子 )
排序后的数组：Array ( [d] => 梨子 [c] => 苹果 [b] => 香蕉 [a] => 柠檬 )
```

8. shuffle -- 将数组打乱

```
<meta http-equiv="content-type" content="text/html;charset=utf-8" />

<?php

$fruits= array('a'=>'柠檬','b'=>'香蕉','c'=>'苹果','d'=>'梨子');

echo '原始的数组：';

print_r($fruits);

echo '<br/>';

shuffle($fruits);

echo '打乱后的数组：';

print_r($fruits);

?>
```

output

```
原始的数组：Array ( [a] => 柠檬 [b] => 香蕉 [c] => 苹果 [d] => 梨子 )
打乱后的数组：Array ( [0] => 香蕉 [1] => 苹果 [2] => 柠檬 [3] => 梨子 )
```

9. array_reverse -- 返回一个单元顺序相反的数组

```
<meta http-equiv="content-type" content="text/html;charset=utf-8" />

<?php

$fruits = array('a'=>'柠檬','b'=>'香蕉','c'=>'苹果','d'=>'梨子');

echo '原始的数组：';

print_r($fruits);

echo '<br/>';

$fruits = array_reverse($fruits);

echo '反转后的数组：';

print_r($fruits);

?>
```

output

```
原始的数组：Array ( [a] => 柠檬 [b] => 香蕉 [c] => 苹果 [d] => 梨子 )
打乱后的数组：Array ( [0] => 香蕉 [1] => 苹果 [2] => 柠檬 [3] => 梨子 )
```

10. array_unshift -- 在数组开头插入一个或多个单元

```
<meta http-equiv="content-type" content="text/html;charset=utf-8" />

<?php

$fruits = array('a'=>'柠檬','b'=>'香蕉','c'=>'苹果','d'=>'梨子');

echo '原始的数组：';

print_r($fruits);

echo '<br/>';

array_unshift($fruits,'柿子');

echo '插入后的数组：';

print_r($fruits);

?>
```

output

```
原始的数组：Array ( [a] => 柠檬 [b] => 香蕉 [c] => 苹果 [d] => 梨子 )
插入后的数组：Array ( [0] => 柿子 [a] => 柠檬 [b] => 香蕉 [c] => 苹果 [d] => 梨子 )
```

11. array_shift -- 将数组开头的单元移出数组

```
<meta http-equiv="content-type" content="text/html;charset=utf-8" />

<?php

$fruits = array('a'=>'柠檬','b'=>'香蕉','c'=>'苹果','d'=>'梨子');

echo '原始的数组：';

print_r($fruits);

echo '<br/>';

array_shift($fruits);

echo '移出后的数组：';

print_r($fruits);

?>
```

output

```
原始的数组：Array ( [a] => 柠檬 [b] => 香蕉 [c] => 苹果 [d] => 梨子 )
移出后的数组：Array ( [b] => 香蕉 [c] => 苹果 [d] => 梨子 )
```

12. array_rand -- 从数组中随机取出一个或多个单元

```
<meta http-equiv="content-type" content="text/html;charset=utf-8" />

<?php

$fruits = array ('柠檬', '香蕉', '苹果', '梨子' );

echo '原始的数组：';

print_r ( $fruits );

echo '<br/>';

$newArr_key = array_rand ( $fruits, 2 );

echo '随机后的数组：';

echo $fruits [$newArr_key [0]].' ';

echo $fruits [$newArr_key [1]];

?>
```

output

```
原始的数组：Array ( [0] => 柠檬 [1] => 香蕉 [2] => 苹果 [3] => 梨子 )
随机后的数组：梨子 苹果
```

13. array_pop -- 将数组最后一个单元弹出（出栈）

```
<meta http-equiv="content-type" content="text/html;charset=utf-8" />

<?php

$fruits = array ('柠檬', '香蕉', '苹果', '梨子' );

echo '原始的数组：';

print_r ( $fruits );

echo '<br/>';

array_pop ( $fruits );

echo '弹出后的数组：';

print_r ( $fruits );

?>
```

Output:

```
原始的数组：Array ( [0] => 柠檬 [1] => 香蕉 [2] => 苹果 [3] => 梨子 )
弹出后的数组：Array ( [0] => 柠檬 [1] => 香蕉 [2] => 苹果 )
```

14. array_push -- 将一个或多个单元压入数组的末尾（入栈）

```
<meta http-equiv="content-type" content="text/html;charset=utf-8" />

<?php

$fruits = array ('柠檬', '香蕉', '苹果', '梨子' );

echo '原始的数组：';

print_r ( $fruits );

echo '<br/>';

array_push ( $fruits,'柿子');

echo '弹出后的数组：';

print_r ( $fruits );

?>
```

Output:

```
原始的数组：Array ( [0] => 柠檬 [1] => 香蕉 [2] => 苹果 [3] => 梨子 )
弹出后的数组：Array ( [0] => 柠檬 [1] => 香蕉 [2] => 苹果 [3] => 梨子 [4] => 柿子 )
```

五 · 数组的指针的操作

each -- 返回数组中当前的键／值对并将数组指针向前移动一步

current -- 返回数组中的当前单元

reset -- 将数组的内部指针指向第一个单元

end -- 将数组的内部指针指向最后一个单元

next -- 将数组中的内部指针向前移动一位

pos -- current() 的别名

prev -- 将数组的内部指针倒回一位

```
<meta http-equiv="content-type" content="text/html;charset=utf-8" />
<?php
$fruits = array ('柠檬', '香蕉', '苹果', '梨子' );
print_r ( $fruits );
echo '<br/>';

echo 'each() : ';
print_r ( each ( $fruits ) );
echo '<br/>';

echo 'current() : ';
echo (current ( $fruits ));
echo '<br/>';

echo 'next() : ';
echo (next ( $fruits ));
echo '<br/>';

echo 'end() : ';
echo (end ( $fruits ));
echo '<br/>';

echo 'prev() : ';
echo (prev ( $fruits ));
echo '<br/>';

echo 'pos() : ';
echo (pos ( $fruits ));
echo '<br/>';
?>
```

Output:

```
Array ( [0] => 柠檬 [1] => 香蕉 [2] => 苹果 [3] => 梨子 )
each() : Array ( [1] => 柠檬 [value] => 柠檬 [0] => 0 [key] => 0 )
current() : 香蕉
next() : 苹果
end() : 梨子
prev() : 苹果
pos() : 苹果
```

六．统计数组个数

count -- 计算数组中的单元数目或对象中的属性个数

sizeof -- count() 的别名

array_count_values -- 统计数组中所有的值出现的次数

```
<?php

$nums = array (1, 3, 5, 1, 3, 4, 5, 65, 4, 2, 2, 1, 4, 4, 1, 1, 4, 1, 5, 4, 5, 4 );


echo count ( $nums );

echo '<br>';


echo sizeof ( $nums );

echo '<br>';


$arrayCount = array_count_values ( $nums );

print_r ( $arrayCount );

?>
```

output

```
22
22
Array ( [1] => 6 [3] => 2 [5] => 4 [4] => 7 [65] => 1 [2] => 2 )
```

七．将数组转换成标量变量：extract()

把数组中的每个元素转换成变量，变量名是数组元素的key,变量值为数组元素的value.

```
<?php

$fruits = array('a'=>'apple','b'=>'banana','o'=>'orange');

extract($fruits);


echo $a.'<br>';

echo $b.'<br>';

echo $o.'<br>';

?>
```

output

```
apple
banana
orange
```

第六章 目录与文件

一．目录操作

basename -- 返回路径中的文件名部分

dirname -- 返回路径中的目录部分

pathinfo -- 返回文件路径的信息

realpath -- 返回规范化的绝对路径名

```
<?php
$path = 'demo1.php';

$path = realpath($path);

echo basename($path);
echo '<br>';

echo dirname($path);
echo '<br>';

$array_path = pathinfo($path);
echo 'basename : '.$array_path['basename'].'<br>';
echo 'dirname : '.$array_path['dirname'].'<br>';
echo 'extension : '.$array_path['extension'].'<br>';
echo 'filename : '.$array_path['filename'].'<br>';
?>
```

Output:

```
demo1.php
D:\AppServ\www\Basic6
basename : demo1.php
dirname : D:\AppServ\www\Basic6
extension : php
filename : demo1
```

二．磁盘、目录和文件计数

1.查看文件大小和磁盘空间

filesize -- 取得文件大小

disk_free_space -- 返回目录中的可用空间

disk_total_space -- 返回一个目录的磁盘总大小

```
<?php
$path ='demo2.php';


$path = realpath($path);


$drive = 'c:';


echo round(filesize($path)/1024,2).'kb'.'<br/>';


echo round(disk_free_space($drive)/1024/1024/1024,2).'GB'.'<br/>';


echo round(disk_total_space($drive)/1024/1024/1024,2).'GB'.'<br/>';
?>
```

output

```
0.26kb
10.61GB
30.01GB
```

2.获得文件的各种时间

fileatime -- 取得文件的上次访问时间

filectime -- 取得文件的 inode 修改时间

filemtime -- 取得文件修改时间

```
<?php

$file = realpath ( '../Basic5/demo3.php' );

$date_format = 'Y-m-d h:i:s';

echo 'lastest accessing time : '.date ( $date_format, fileatime ( $file ) ) . '<br>';

echo 'lastest change time : '.date ( $date_format, filectime ( $file ) ) . '<br>';

echo 'lastest modify time : '.date ( $date_format, filemtime ( $file ) ) . '<br>';

?>
```

output

```
lastest accessing time : 2011-12-18 04:26:49
lastest change time : 2011-12-18 04:26:49
lastest modify time : 2011-12-18 04:29:15
```

三・文件处理

文件读写的两种方式：

1.php所有版本都支持的方法：

fopen -- 打开文件或者 URL

fclose -- 关闭一个已打开的文件指针

fwrite -- 写入文件（可安全用于二进制文件）

表 1. fopen() 中 mode的可能值列表

mode	说明
'r'	只读方式打开，将文件指针指向文件头。
'r+'	读写方式打开，将文件指针指向文件头。
'w'	写入方式打开，将文件指针指向文件头并将文件大小截为零。如果文件不存在则尝试创建之。
'w+'	读写方式打开，将文件指针指向文件头并将文件大小截为零。如果文件不存在则尝试创建之。
'a'	写入方式打开，将文件指针指向文件末尾。如果文件不存在则尝试创建之。
'a+'	读写方式打开，将文件指针指向文件末尾。如果文件不存在则尝试创建之。
'x'	创建并以写入方式打开，将文件指针指向文件头。如果文件已存在，则 `fopen()` 调用失败并返回 `FALSE`，并生成一条 `E_WARNING` 级别的错误信息。如果文件不存在则尝试创建之。这和给 底层的 `open(2)` 系统调用指定 `O_EXCL|O_CREAT` 标记是等价的。此选项被 PHP 4.3.2 以及以后的版本所支持，仅能用于本地文件。
'x+'	创建并以读写方式打开，将文件指针指向文件头。如果文件已存在，则 `fopen()` 调用失败并返回 `FALSE`，并生成一条 `E_WARNING` 级别的错误信息。如果文件不存在则尝试创建之。这和给 底层的 `open(2)` 系统调用指定 `O_EXCL|O_CREAT` 标记是等价的。此选项被 PHP 4.3.2 以及以后的版本所支持，仅能用于本地文件。

```
<?php

$fp = fopen('file1.txt','w');

$outStr = "my name is anllin,\r\nmy age is 29.";

fwrite($fp,$outStr,strlen($outStr));

fclose($fp);

?>
```

output

```
my name is anllin,

my age is 29.
```

2.php5新加入的方法

file_put_contents -- 将一个字符串写入文件

```
<?php
file_put_contents('file2.txt',"my name is anllin,\r\nmy age is 29.");
?>
```

output

```
my name is anllin,
my age is 29.
```

读出文件内容的方法：

函数	功能
Fgetc()	读出一个字符，并将指针移到下一个字符
Fgets()	读出一行字符，可以指定一行显示的长度。
Fgetss()	从文件指针中读取一行并过滤掉HTML标记
Fread()	读取定量的字符
Fpassthru()	输出文件到指定处的所有剩余数据
File()	将整个文件读入数组中，以行分组
Readfile()	读入一个文件并写入到输出缓冲
File_get_contents()	将整个文件读入一个字符串
Feof()	判断读完文件函数
File_exists()	查看文件是否存在

示例文件file1.txt的内容如下：

```
my name is anllin,
my age is 29.
```

fgetc -- 从文件指针中读取字符

Demo.php

```
<?php
$fp = fopen('file1.txt','r');
echo fgetc($fp);
echo fgetc($fp);
fclose($fp);
?>
```

Output:

```
my
```

fgets -- 从文件指针中读取一行

```
<?php
$fp = fopen('file1.txt','r');
echo fgets($fp);
echo fgets($fp);
fclose($fp);
?>
```

output

```
my name is anllin, my age is 29.
```

fgetss -- 从文件指针中读取一行并过滤掉 HTML 标记

```
<?php
$fp = fopen('file3.txt','w');
$outStr = "my name is <b>anllin</b>";
fwrite($fp,$outStr,strlen($outStr));
fclose($fp);

$ftp = fopen('file3.txt','r');
echo fgetss($ftp);
fclose($ftp);
?>
```

Output

```
my name is anllin
```

fread -- 读取文件（可安全用于二进制文件）

```
<?php
$filename = 'file1.txt';
$fp = fopen($filename,'r');
$contents = fread($fp,filesize($filename));
echo $contents;
fclose($fp);
?>
```

Output

```
my name is anllin, my age is 29.
```

fpassthru -- 输出文件指针处的所有剩余数据

```
<?php
$filename = 'file1.txt';
$fp = fopen($filename,'rb');
$leftSize = fpassthru($fp);
echo $leftSize;
fclose($fp);
?>
```

output

```
my name is anllin, my age is 29.  33
```

file -- 把整个文件读入一个数组中

```
<?php
$lines = file('file1.txt');
foreach ($lines as $line_num => $line)
{
    echo $line_num.' : '.$line.'<br>';
}
?>
```

output

```
0 : my name is anllin,
1 : my age is 29.
```

readfile -- 输出一个文件

```
<?php
$size = readfile('file1.txt');
echo $size;
?>
```

output

```
my name is anllin, my age is 29.33
```

file_get_contents -- 将整个文件读入一个字符串(php5.0新增)

```
<?php
$contents = file_get_contents('file1.txt');
echo $contents;
?>
```

output

```
my name is anllin, my age is 29.
```

feof -- 测试文件指针是否到了文件结束的位置

```
<?php
$fp = fopen('file1.txt','r');
while(!feof($fp))
{
    echo fgetc($fp);
}
fclose($fp);
?>
```

output

```
my name is anllin, my age is 29.
```

file_exists -- 检查文件或目录是否存在

```
<meta http-equiv="content-type" content="text/html;charset=utf-8"/>
<?php
$filename = 'file1.txt';
if(file_exists($filename))
{
    echo '执行文件读写操作';
}
else
{
    echo '你要找的文件不存在';
}
?>
```

output

```
执行文件读写操作
```

filesize -- 取得文件大小

```
<?php
$file_size = filesize('file1.txt');
echo $file_size;
?>
```

output

```
33
```

unlink -- 删除文件

```
<?php
$isDeleted = unlink('file3.txt');
echo $isDeleted;
?>
```

output

```
1
```

rewind -- 倒回文件指针的位置

ftell -- 返回文件指针读/写的位置

fseek -- 在文件指针中定位

```
<?php
$fp = fopen ( 'file1.txt', 'r' );
fgetc ( $fp );
fgetc ( $fp );
echo ftell ( $fp ) . '<br>';
rewind ( $fp );
echo ftell ( $fp ) . '<br>';
fgetc ( $fp );
fgetc ( $fp );
echo ftell ( $fp ) . '<br>';
fseek($fp,0);//same as rewind ($fp)
echo ftell ( $fp ) . '<br>';
?>
```

output

```
2020
```

Flock的操作值

操作值	意义
LOCK_SH(以前为1)	读写锁定。这意味着文件可以共享，其他人可以读该文件
LOCK_EX(以前为2)	写操作锁定。这是互斥的，该文件不能被共享
LOCK_UN(以前为3)	释放已有的锁定
LOCK_NB(以前为4)	防止在请求加锁时发生阻塞

flock -- 轻便的咨询文件锁定

```
<?php
$filename = 'file1.txt';
$fp = fopen($filename,'rb');
flock($fp,LOCK_EX);//locked
$contents = fread($fp,filesize($filename));
flock($fp,LOCK_UN);//unlocked
echo $contents;
fclose($fp);
?>
```

output

```
my name is anllin, my age is 29.
```

目录句柄操作

opendir -- 打开目录句柄

readdir -- 从目录句柄中读取条目

closedir -- 关闭目录句柄

```
<?php
$dir= opendir('../Basic6');
while(!!$file = readdir($dir))
{
    echo $file.'<br/>';
}
closedir($dir);
?>
```

output

```
.
..
.buildpath
.project
.settings
demo1.php
demo10.php
demo11.php
demo12.php
demo13.php
demo14.php
demo15.php
demo16.php
demo17.php
demo18.php
demo19.php
demo2.php
demo20.php
demo3.php
demo4.php
demo5.php
demo6.php
demo7.php
demo8.php
demo9.php
file1.txt
file2.txt
```

scandir -- 列出指定路径中的文件和目录

```
<?php

$files = scandir('../Basic6');

foreach($files as $file)

{

    echo $file.'<br>';

}

?>
```

output

```
.
..
.buildpath
.project
.settings
demo1.php
demo10.php
demo11.php
demo12.php
demo13.php
demo14.php
demo15.php
demo16.php
demo17.php
demo18.php
demo19.php
demo2.php
demo20.php
demo21.php
demo3.php
demo4.php
demo5.php
demo6.php
demo7.php
demo8.php
demo9.php
file1.txt
file2.txt
```

rename -- 重命名一个文件或目录

```
<?php

rename('demo1.php','demo01.php');

if(file_exists('demo01.php'))

{

    echo 'file rename success';

}

else

{

    echo 'file rename fail';

}

?>
```

output

```
file rename success
```

rmdir -- 删除目录

```
<?php
rmdir('123');
if(file_exists('123'))
{
    echo 'delete file fail';
}
{
    echo 'delete file success';
}
?>
```

output

```
delete file success
```

第七章 自定义函数

使用自定义函数的目的：避免大量重复代码的出现。

7.1 · 标准函数

标准php发行包中有1000多个标准函数，这些标准函数都是系统内置的，不需要用户创建就可以直接使用

如：

```
<?php

echo md5('123456');

echo '<br/>';

echo sha1('123456');

echo '<br/>';

echo pi();

?>
```

output

```
e10adc3949ba59abbe56e057f20f883e
7c4a8d09ca3762af61e59520943dc26494f8941b
3.14159265359
```

7.2 · 自定义函数

7.2.1函数命名基本原则：

1.函数名不能和已有的函数名重名。

2.函数名只能包含字母、数字和下划线。

3.函数名不能以数字开头

7.2.2基本使用：用function进行声明

```
<?php
//创建函数
function funcCountArea($radius)
{
    return $radius*$radius*pi();
}

//使用函数
$area = funcCountArea(20);
echo $area;
echo '<br/>';

$area2 = funcCountArea(30);
echo $area2;
?>
```

output

```
1256.63706144
2827.43338823
```

7.2.3按值传参

```
<?php
$a = 5;
function funcChange($a)
{
    $a = 2 * $a;
}
funcChange ($a);
echo $a;
?>
```

output

```
5
```

7.2.4按引用传参

```
<?php
$a = 5;
function funcChange(&$a)
{
    $a = 2 * $a;
}
funcChange ($a);
echo $a;
?>
```

output

```
10
```

7.2.5返回多个值的函数调用

```
<?php
function funcUserInfo($username,$password)
{
    $userInfo = array($username,$password);
    return $userInfo;
}

$arr = funcUserInfo('anllin','123456');

print_r($arr);
?>
```

output

```
Array ( [0] => anllin [1] => 123456 )
```

7.2.6另一种返回多个值的函数调用（实用：推荐）

```
<?php
function funcUserInfo($username, $password)
{
    $userInfo [] = $username;
    $userInfo [] = $password;
    return $userInfo;
}

$arr[] = funcUserInfo ( 'Bob', '512655' );
$arr[] = funcUserInfo ( 'John', '458736' );
$arr[] = funcUserInfo ( 'Mark', '925472' );
print_r ( $arr );
?>
```

output

```
Array ( [0] => Array ( [0] => Bob [1] => 512655 ) [1] => Array ( [0] => John [1] => 45873
```

注意：函数调用是不区分大小写的，但是变量名是区分大小写的。

7.2.7理解作用域：

局部变量:

在函数内部声明的变量。

全局变量：

在函数外部声明的变量。

7.2.8局部变量转换成全局变量

```
<?php
$a = 5;
function funcChangeValue()
{
    global $a;
    $a = 10;
}

funcChangeValue();

echo $a;
?>
```

output

```
10
```

7.2.9超级全局变量$GLOBALR的使用

```
<?php
$GLOBALS['a'] = 5;
function funcChangeValue()
{
    $GLOBALS['a'] = 10;
}

funcChangeValue();

echo $GLOBALS['a'];
?>
```

Output

```
10
```

7.3 · 文件包含

7.3.1 Include的使用，可以包含相同的文件多次

```
<?php

include 'demo1.php';

include 'demo1.php';

include 'demo1.php';

?>
```

output

```

e10adc3949ba59abbe56e057f20f883e
7c4a8d09ca3762af61e59520943dc26494f8941b
3.14159265359

e10adc3949ba59abbe56e057f20f883e
7c4a8d09ca3762af61e59520943dc26494f8941b
3.14159265359

e10adc3949ba59abbe56e057f20f883e
7c4a8d09ca3762af61e59520943dc26494f8941b
3.14159265359
```

7.3.2 include_once使用上和include没什么区别，但是调用多次只会包含相同的文件一次

```
<?php

include_once 'demo1.php';

include_once 'demo1.php';

include_once 'demo1.php';

?>
```

output

```

e10adc3949ba59abbe56e057f20f883e
7c4a8d09ca3762af61e59520943dc26494f8941b
3.14159265359s
```

7.3.3 require() 语句包含并运行指定文件。

```

<?php

require 'demo1.php';

require 'demo1.php';

require 'demo1.php';

?>
```

output

```
e10adc3949ba59abbe56e057f20f883e
7c4a8d09ca3762af61e59520943dc26494f8941b
3.14159265359

e10adc3949ba59abbe56e057f20f883e
7c4a8d09ca3762af61e59520943dc26494f8941b
3.14159265359

e10adc3949ba59abbe56e057f20f883e
7c4a8d09ca3762af61e59520943dc26494f8941b
3.14159265359
```

7.3.4 require_once() 语句在脚本执行期间包含并运行指定文件.但是不重复包含相同的文件。

```
<?php

require_once 'demo1.php';

require_once 'demo1.php';

require_once 'demo1.php';

?>
```

output

```
e10adc3949ba59abbe56e057f20f883e
7c4a8d09ca3762af61e59520943dc26494f8941b
3.14159265359s
```

7.3.5 include与require的区别

Include后面如果还有其他代码，当调用include出错时，后面的代码还会继续执行，但是require则不会。

Include在调用一个不存在的文件时，会给出警告，但是会继续执行后面的代码。

```
<?php

include 'demo111.php';


echo('this is demo13.php');

?>
```

output

```
Warning: include(demo111.php) [function.include]: failed to open stream: No such file or

Warning: include() [function.include]: Failed opening 'demo111.php' for inclusion (includ
this is demo13.php
```

Require在调用一个不存在的文件时，会给出一个错误，并中止代码的执行。

```
<?php

require 'demo111.php';


echo('this is demo14.php');

?>
```

Output

```
Warning: require(demo111.php) [function.require]: failed to open stream: No such file or

Fatal error: require() [function.require]: Failed opening required 'demo111.php' (include
```

7.4 · 魔法常量

名称	描述
__FILE__	当前文件名
__LINE__	当前行号
__FUNCTION__	当前函数名
__CLASS__	当前类名
__METHOD__	当前方法名

所谓的魔法常量，并不是眞的常量，而是根据场合去获取固定值的变量

```
<?php

echo __FILE__;

echo '<br>';


echo __LINE__;

echo '<br>';


function funcTest()

{

    echo __FUNCTION__;

}

funcTest();

?>
```

output

```
D:\AppServ\www\Basic7\demo15.php
5
funcTest
```

第八章 字符串处理

学习要点：

1.字符串格式化

2.操作子字符串

3.字符串比较

4.查找替换字符串

5.处理中文字符

在每天的编程工作中，处理、调整以至最后控制字符串是很重要的一部分，一般也认为

这是所有编程语言的基础。不同与其他语言，PHP 没有那么麻烦地使用数据类型来处理字

符串。这样一来，PHP 中的字符串处理就再容易不过了。

一．字符串格式化

整理字符串的第一步是清理字符串中多余的空格。虽然这一部操作不是必需的，但如果

要将字符串存入一个文件或数据库中，或者将它和别的字符串进行比较，这就是非常有用
的。

chop()函数移除字符串后面多余的空白，包括新行。

ltrim()函数移除字符串起始处多余空白。

rtrim()函数移除字符串后面多余的空白，包括新行，此函数是chop()的别名。

trim()函数移除字符串两边多余的空白。

```
<?php echo trim('        PHP       '); ?>
```

PHP 具有一系列可供使用的函数来重新格式化字符串，这些函数的工作方式是各不相

同的。

nl2br()函数将字符串作为输入参数，用XHTML 中的
标记代替字符串中的换行符。

```
<?php echo nl2br("This is a Teacher!\nThis is a Student!"); ?>
```

将特殊字符转换为HTML 等价形式，可以使用htmlentities()和htmlspecialchars 函数。

如果想要去掉字符串中的HTML 去掉，可以使用strip_tags()函数

```
<?php echo htmlentities('<strong>我是吴祁！</strong>'); //转换所有字符
echo htmlspecialchars('<strong>我是吴祁！</strong>') //转换特殊字符
echo strip_tags('<strong>我是吴祁！</strong>') //去掉了<strong>
?>
```

对于字符串来说，某些字符肯定是有效的，但是当将数据插入到数据库中的时候可能会引起一些问题，因为数据库会将这些字符解释成控制符。这些有问题的字符就是引号（单引号和双引）、反斜杠（\）和NULL 字符。

PHP 提供了两个专门用于转义字符串的函数。在将任何字符串写到数据库之前，应该使用addslashes()将它们重新格式化，

在调用了addslashes()后，所有的引号都加了斜杠，而stripslashes()函数去掉了这些斜杠。

```
<?php echo addslashes('This is \a" Teacher! '); ?>
```

可以重新格式化字符串中的字母大小写。

strtoupper()函数将字符串转换为大写

strtolower()函数将字符串转换成小写

ucfirst()函数将第一个字母转换为大写

ucwords()函数将每个单词第一个字母转换为大写

```
<?php echo strtoupper('yc60.com@gmail.com'); ?>
```

填充字符串函数：str_pad()将字符串用指定个数的字符填充字符串。

```
<?php echo str_pad('Salad',10).'is good.'; ?>
```

二 · 操作子字符串

通常，我们想查看字符串的各个部分。例如，查看句子中的单词，或者将一个域名或电子邮件地址分割成一个个的组件部分。PHP 提供了几个字符串函数来实现此功能。

使用函数explode()、implode()和join()

为了实现这个功能，我们将使用的第一个函数是explode()。

使用implode()或join()函数来实现与函数explode()相反的效果，这两个函数的效果是一

致的。

```
<?php $email = 'yc60.com@gmail.com'; $email_array = explode('@',$email); ?>
```

使用strtok()函数

strtok()函数一次只从字符串取出一些片段（称为令牌）。对于一次从字符串中取出一个

单词的处理来说，strtok()函数比explode()函数的效果更好。

```
<?php $str = "I,will.be#back"; $tok = strtok($str,",.#"); while($tok) {     echo "$tok<br
    $tok = strtok(",.#");
} ?>
```

使用substr()函数

函数substr()允许我们访问一个字符串给定起点和终点的子字符串。这个函数并不适用

于我们的例子中，但是，当需要得到某个固定格式字符串中的一部分时，它会非常有用。

```
<?php echo substr("abcdef", 1, 3); ?>
```

分解字符串：str_split()返回一个数组，其中各数组元素分别是字符串参数中的一个字符

串。

```
<?php print_r(str_split('This is a Teacher!')); ?>
```

逆置字符串：strrev()可以将一个字符串逆反过来。

```
<?php echo strrev('This is a Teacher!'); ?>
```

三．字符串比较

到目前为止，我们已经用过"= ="号来比较两个字符串是否相等。使用PHP 可以进行一

些更复杂的比较。这些比较分为两类：部分匹配和其他情况。

字符串的排序：strcmp()、strcasecmp()和strnatcmp()

该函数需要两个进行比较的参数字符串。如果这两个字符串相等，该函数返回0，如果

按字典顺序str1 和str2 后面（大于str2）就返回一个正数，如果str1 小于str2 就返回一个负

数。这个函数是区分大小写的。

函数strcasecmp()除了不区分大小写之外，其他和strcmp()一样。

函数strnatcmp()及与之对应的不区分大小写的strnatcasecmp()函数是在PHP4 中新增的。

这两个函数按“自然排序”比较字符串，所谓自然排序是按人们习惯的顺序进行排序。

```
<?php echo strcmp('a','b'); ?>
```

使用strspn()函数返回一个字符串中包含有另一个字符串中字符的第一部分的长度。也

就是求两个字符串之间相同的部分。

```
<?php echo strspn('gmail','yc60.com@gmail.com'); ?>
```

使用strlen()函数测试字符串的长度

可以使用函数strlen()来检查字符串的长度。如果传给它一个字符串，这个函数将返回

字符串的长度。例如, strlen("hello") 将返回5.

```
<?php echo strlen('This is a Teacher!'); ?>
```

确定字符串出现的频率：substr_count()返回一个字符串在另一个字符串中出现的次数。

```
<?php echo substr_count('yc60.com@gmail.com','c'); ?>
```

四．查找替换字符串

通常，我们需要检查一个更长的字符串中是否含有一个特定的子字符串。这种部分匹配

通常比测试字符串的完全等价更有用处。

在字符串中查找字符串：strstr()、strchr()、strrchr()和stristr()

函数strstr()是最常见的，它可以用于在一个较长的字符串专供查找匹配的字符串或字

符。请注意，函数strchr()和strstr()完全一样。

```
<?php echo strstr('yc60.com@gmail.com','@'); ?>
```

函数strstr()有两个变体。第一个变体是stristr()，它几乎和strstr()一样，其区别在于不区

分字符大小。对于我们的只能表单应用程序来说，这个函数非常有用，因为用户可以输入

"delivery"、"Delivery"和"DELIVERY"。

第二个变体是strrchr()，它也几乎和strstr()一样，只不过是strstr()的别名。

查找字符串的位置：strpos()、strrpos()。

函数strpos()和strrpos()的操作和strstr()类似，但它不是返回一个子字符串，而返回子字符串needle 在字符串haystack 中的位置。更有趣的是，现在的PHP 手册建议使用strpos()函数代替strstr()函数来查看一个子字符串在一个字符串中出现的位置，因为前者的运行速度更快。

```
<?php echo strrpos('yc60.com@gmail.com','c'); ?>
```

替换字符串：str_replace()、str_ireplace()、substr_replace()

```
<?php echo str_replace('@','#','yc60.com@gmail.com'); echo substr_replace('yc60.com@gmail
```

五.处理中文字符

对于以上的字符串函数，有些可以用于中文，但有些却不适用中文。所以，PHP 提供了专门的函数来解决这样的问题。

中文字符可以是gbk,utf8,gb2312

mb_strlen() 对应的函数为strlen() 求字符串的长度

mb_strstr() 对应的函数为strstr() 求某字符串到结尾的字符

mb_strpos() 对应的函数为strpos() 求出字符最先出现处

mb_substr() 对应的函数为substr() 取出指定的字符串

mb_substr_count() 对应函数为substr_str() 返回字符串出现的次数

最后扫一遍帮助手册

注：文章出自李炎恢**PHP**视频教程，本文仅限交流使用，不得用于商业用途，否则后果自负。

第九章 正则表达式

学习要点：

1.正则表达式语法（Perl 风格）

2.正则表达式中的元素

3.Perl 风格函数

处理字符串时，有很多较为复杂的字符串用普通的字符串处理函数无法干净的完成。比如说，可能需要验证一个Email 地址是否合法，为此需要查看许多不容易检查的规则。这正是正则表达式的用武之地。正则表达式是功能强大而简明的字符组，其中可以包含大量的逻辑，特别值得一提的是正则表达式相当简短。

一．正则表达式语法（**Perl**风格）

Perl 一直被认为是最伟大的解析语言之一，它提供了一种全面的正则表达式，即使是最复杂的字符串模式，也可以用这种正则表达式语言搜索和替换。PHP 开发人员认识到，与其重新发明正则表达式，不如让PHP 用户直接使用声名赫赫的Perl 正则表达式语言，即Perl 风格的函数。

模式规则：/php/ 在字符串前后加上两条斜杠即可。

匹配函数：preg_match()函数在字符串中搜索模式，如果存在则返回true，否则返回false。

```
<?php preg_match('/php/','php'); ?>
```

二．正则表达式中的元素

正则表达式中包含三种元素分别为：量词、元字符、修饰符

量词

<table>
<tr><th>语法</th><th>描述</th></tr>
<tr><td>+</td><td>匹配任何至少包含一个前导字符串</td></tr>
<tr><td>*</td><td>匹配任何包含零个或多个前导字符串</td></tr>
<tr><td>?</td><td>匹配任何包含零个或一个前导字符串</td></tr>
<tr><td>.</td><td>匹配任意字符串</td></tr>
<tr><td>{x}</td><td>匹配任何包含 x 个前导字符串</td></tr>
<tr><td>{x,y}</td><td>匹配任何包含 x 到 y 个前导字符串</td></tr>
<tr><td>{x,}</td><td>匹配任何包含至少 x 个前导字符串</td></tr>
<tr><td>$</td><td>匹配字符串的行尾</td></tr>
<tr><td>^</td><td>匹配字符串的行首</td></tr>
<tr><td>|</td><td>匹配字符串的左边或者右边</td></tr>
<tr><td>()</td><td>包围一个字符分组或定义个反引用，可以使用\1\2 提取</td></tr>
</table>

元字符

语法	描述
[a-z]	匹配任何包含小写字母 a-z 的字符串
[A-Z]	匹配任何包含大写字母 A-Z 的字符串
[0-9]	匹配任何包含数字 0-9 的字符串
[abc]	匹配任何包含小写字母 a、b、c 的字符串
[^abc]	匹配任何不包含小写字母 a、b、c 的字符串
[a-zA-Z0-9_]	匹配任何包含 a-zA-Z0-9 和下划线的字符串
\w	匹配任何包含 a-zA-Z0-9 和下划线的字符串（同上）
\W	匹配任何没有下划线和字母数字的字符串
\d	匹配任何数字字符，和[0-9]相同
\D	匹配任何非数字字符，和[^0-9]相同
\s	匹配任何空白字符
\S	匹配任何非空白字符
\b	匹配是否到达了单词边界
\B	匹配是否没有达到单词边界
\	匹配正则中的特殊字符

修饰符

语法	描述
i	完成不区分大小写的搜索
m	在匹配首内容或者尾内容时候采用多行识别匹配
x	忽略正则中的空白
A	强制从头开始匹配
U	禁止贪婪匹配 只跟踪到最近的一个匹配符并结束

三 · **Perl**风格函数

PHP 为使用Perl 兼容的正则表达式搜索字符串提供了7 个函数，包括：preg_grep()、

preg_match()、preg_match_all()、preg_auote()、preg_replace()、preg_replace_callback()和

preg_split()。

搜索字符串：preg_grep()函数搜索数组中的所有元素，返回由与某个模式匹配的所有元素组成的数组。

```
<?php $language = array('php','asp','jsp','python','ruby'); print_r(preg_grep('/p$/',$lan
```

搜索模式：preg_match()函数在字符串中搜索模式，如果存在则返回true，否则返回false。

```
<?php echo preg_match('/php[1-6]/','php5'); ?>
```

电子邮件验证小案例（分组应用）

```
<?php $mode = '/([\w\.\_]{2,10})@(\w{1,}).([a-z]{2,4})/'; $string = 'yc60.com@gmail.com';
```

匹配模式的所有出现：preg_match_all()函数在字符串中匹配模式的所有出现，然后将所有匹配到的全部放入数组。

```
<?php preg_match_all('/php[1-6]/','php5sdfphp4sdflljkphp3sdlfjphp2',$out); print_r($out);
```

定界特殊的正则表达式：preg_quote()在每个对于正则表达式语法而言有特殊含义的字符前插入一个反斜线。这些特殊字符包含：`$ ^ * () + = {} [] | \\ : <>`。

```
<?php echo preg_quote('PHP的价格是：$150'); ?>
```

替换模式的所有出现：preg_replace()函数搜索到所有匹配，然后替换成想要的字符串返回出来。

```
<?php echo preg_replace('/php[1-6]/','python','This is a php5,This is a php4'); ?>
```

ubb 小案例：贪婪问题+分组使用()

```
<?php $mode = '/\[b\](.*)\[\/b\]/U'; $replace = '<strong>\1</strong>'; $string = 'This is
```

以不区分大小写的方式将字符串划分为不同的元素：preg_split()用来分割不同的元素。

```
<?php print_r(preg_split('/[\.@]/','yc60.com@gmail.com' )); ?>
```

注：目前为PHP 使用POSIX 风格的正则表达式搜索字符串提供了7 个函数，包括：ereg()、ereg_replace()、eregi()、eregi_replace()、split()、spliti()和sql_regcase()。

PS：和Perl 风格基本相同，可以参考手册自行学习。

注：文章出自李炎恢**PHP**视频教程，本文仅限交流使用，不得用于商业用途，否则后果自负。

第十章 日期与时间

学习要点：

1.PHP 日期和时间库

```
使用PHP 编程时，与你遇到的大多数其他类型的数据相比，日期和时间有很大不同。
```

因为日期和时间没有明确的结构，并且日期的计算和表示也很麻烦。在PHP 中，日期和时间函数库是PHP 语言的一个核心部分。

```
时间戳是自1970 年1 月1 日（00:00:00 GMT）以来的秒数。它也被称为Unix 时间
```

戳（Unix Timestamp）。Unix 时间戳(Unix timestamp)，或称Unix 时间(Unix time)、POSIX 时

间(POSIX time)，是一种时间表示方式，定义为从格林威治时间1970 年01 月01 日00 时00 分00 秒起至现在的总秒数。Unix 时间戳不仅被使用在Unix 系统、类Unix 系统中，也在许多其他操作系统中被广泛采用。例如(1184557366 表示2007-07-16 03:42:46)

一．**PHP**日期和时间库

验证日期：**checkdate**()函数能够很好地验证日期，提供的日期如果有效，则返回true，否则返回false。

```
<?php if (checkdate(2,29,2007)) {
 echo '日期合法';
} else { echo '日期不合法';
} ?>
```

格式化时间和日期：**date**()函数返回根据预定义指令格式化时间和日期的字符串形式。所有格式参数，可以参考手册。

```
<?php echo date('Y-m-d H:i:sa'); //直接输入日期和时间
echo date('今天的日期和时间为：Y/m/d H:i:sa'); //可以插入无关的字符串
?>
```

查看更多时间信息：**gettimeofday**()函数返回与当前时间有关的元素所组成的一个关联数组。

```
<?php print_r(gettimeofday()); //可以传入一个真(1)
?>
```

将时间戳转换成友好的值：**getdate()**函数接受一个时间戳，并返回一个由其各部分组成的关联数组。如果不给参数，那么返回当前的时间和日期。

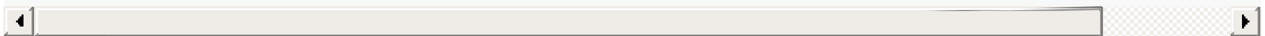
```
<?php print_r(getdate(1184557366)); ?>
```

获取当前的时间戳：**time()**函数可以获取当前的时间戳，并且可以通过设置时间戳的值。

```
<?php echo date('Y-m-d H:i:s',time()+(7*24*60*60)); ?>
```

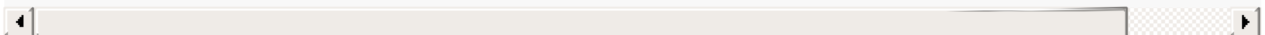
获取特定的时间戳：**mktime()**函数可以生成给定日期时间的时间戳。

```
<?php echo mktime(14,14,14,11,11,2007); echo date('Y-m-d H:i:s',mktime(14,14,14,11,11,200
```



计算时间差

```
<?php $now = time(); $taxday = mktime(0,0,0,7,17,2010); echo round(($taxday - $now)/60/60
```

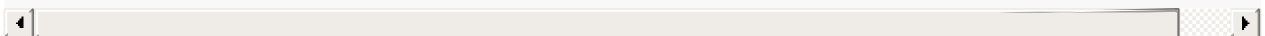


将日期转换成时间戳：**strtotime()**将人可读的日期转换为Unix 时间戳。

```
<?php echo strtotime('2007-10-31 14:31:33'); ?>
```

计算时间差

```
<?php echo (strtotime('2007-10-31 14:31:33') - strtotime('2007-10-31 11:31:33'))/60/60; ?
```



获取当前文件最后修改时间：**getlastmod()**可以得到当前文件最后修改时间的时间戳。

```
<?php echo date('Y-m-d H:i:s',getlastmod()); ?>
```

设置时区和**GMT/UTC**：

修改php.ini 文件中的设置，找到[date]下的;date.timezone = 选项，将该项修改为date.timezone=Asia/Shanghai，然后重新启动apache 服务器。

putenv()函数可以设置当前的默认时区。

```
<?php putenv('TZ=Asia/Shanghai'); echo date('Y-m-d H:i:s'); ?>
```

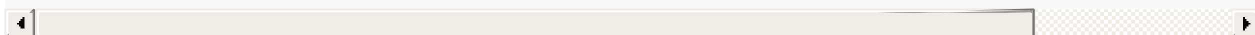
date_default_timezone_set()可以设置当前的默认时区。

date_default_timezone_get()可以获取当前的默认时区。

```
<?php
date_default_timezone_set('Asia/Shanghai'); echo date('Y-m-d H:i:s'); ?>
```

取得本地时间**localtime**()函数可以取得本地时间数据，然后返回一个数组。

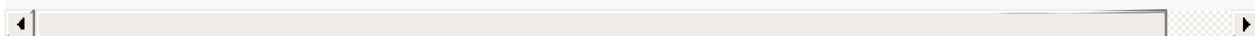
```
<?php
date_default_timezone_set('Asia/Shanghai'); print_r(localtime()); print_r(localtime(time(
```



计算页面脚本运行时间：**microtime**()函数，该函数返回当前UNIX 时间戳和微秒数。返回格式为msec sec 的字符串，其中sec 是当前的UNIX 时间戳，msec 为微秒数。

```
<?php function fntime() { list($msec, $sec) = explode(' ', microtime()); return $msec+$se
} $start_time = fntime(); for($i=0;$i<1000000;$i++) {

} $end_time = fntime(); echo round($end_time - $start_time,4); ?>
```



注：文章出自李炎恢**PHP**视频教程，本文仅限交流使用，不得用于商业用途，否则后果自负。

第十一章 表单与验证

学习要点：

1.Header()函数

2.接收及验证数据

我们对Web 感兴趣，认为它有用的原因是其主要通过基于HTML 的表单发布和收集信

息的能力。这些表单用来鼓励网站的反馈、进行论坛会话、收集在线定购的邮件地址，等等。

但是对HTML 表单进行编码只是有效接受用户输入的必须操作的一部分，必须由服务器端

组件来处理输入。

一．**Header()**函数

标头(header) 是服务器以HTTP 协议传HTML 资料到浏览器前所送出的字符串，在

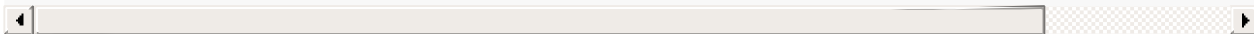
标头与HTML 文件之间尚需空一行分隔。

1.用于重新导向指定的URL

```
<?php header('Location:http://www.baidu.com'); ?>
```

2.用于设置页面字符编码

```
<?php header('Content-Type: text/html; charset=gbk'); echo '嘿嘿，我是中文！页面编码是GBK，文件
```



注意：除非启用了输出缓冲，否则这些命令必须在返回任何输出之前执行。

启用输出缓冲：ob_start()

```
<?php ob_start(); ?>
```

二．接受及验证数据

HTML 表单元素

表单元素	描述
text input	文本框

passoword input	密码框
hidden input	隐藏框
select	下拉列表框
checkbox	复选框
radio	单选按钮
textarea	区域框
file	上传
submit	提交按钮
reset	重置按钮

GET 与POST

处理表单时，必须指定输入到表单的信息以何种方式传输到其目的地（method=""）。对此，Web 开发人员可以采用GET 和POST。使用GET 方法发送数据时，所有域都追加到浏览器的URL 后面，并且为数据随URL 地址发送。采用POST 方法时，值会作为标准值发送。

PHP 分别使用$_GET 和$_POST 超全局变量来处理GET 和POST 变量。通过使用这两个超全局变量，可以准确地指定信息应当来自哪里，并以你希望的方式处理数据。

使用$_GET 或$_POST 来接收数据

1.$_GET['username']，发送的表单method 必须是get；

2.$_POST['username']，发送的表单method 必须是post；

3.采用isset()来验证$_GET['username']超级全局变量是否定义；

4.使用htmlspecialchars()函数将HTML 特殊字符进行过滤。

对数据有效性进行验证

1.使用函数trim()去除数据的前后空格；

2.使用函数strlen()判断数据的长度；

3.使用函数is_numeric()判断数据是纯数字；

4.使用正则表达式验证邮箱是否合法。

```
<?php if (!isset($_POST['send']) || $_POST['send']!='提交') { header('Location:Demo1.php')
} if (preg_match('/([\w\.]{2,255})@([\w\-]{1,255}).([a-z]{2,4})/',$_POST['email'])) { ech
} else { echo '电子邮件不合法';
} ?>
```

注：文章出自李炎恢**PHP**视频教程，本文仅限交流使用，不得用于商业用途，否则后果自负。

第十二章 会话控制

学习要点：

1.Cookie 的应用

2.Session 会话处理

```
HTTP（超文本传输协议）定义了通过万维网（WWW）传输文本、图形、视频和所有
```

其他数据所有的规则。HTTP 是一种无状态的协议，说明每次请求的处理都与之前或之后的请求无关。虽然这种简化实现对于HTTP 的普及做出了卓越的贡献，但对于希望创建复杂的Web 应用程序的开发人员来说，这点有点困扰。为了解决这个问题，出现了一种在客户端机器上存储少量信息（cookie）。

由于cookie 大小限制、数量及其他原因，开发人员又提出了一种解决方案：session 会话处理。

一．**Cookie**的应用

设置cookie：**setcookie**()函数可以在客户端生成一个cookie 文件，这个文件可以保存到期时间、名称、值等。

创建cookie

```
<?php setcookie('name','Lee',time()+(7*24*60*60)); ?>
```

参数1：cookie 名称

参数2：cookie 值

参数3：cookie 过期时间

读取cookie

```
<?php echo $_COOKIE['name']; ?>
```

删除cookie

```
<?php setcookie('name',''); setcookie('name','Lee',time()-1); ?>
```

使用Cookie 的限制

1、必须在HTML 文件的内容输出之前设置；

2、不同的浏览器对Cookie 的处理不一致，且有时会出现错误的结果。

3、限制是在客户端的。一个浏览器能创建的Cookie 数量最多为30 个，并且每个不能超过4KB，每个WEB 站点能设置的Cookie 总数不能超过20 个。

二．**Session**会话处理

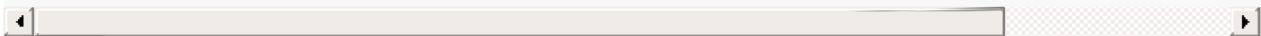
在使用session 会话处理，必须开始session，使用session_start()开始会话。

创建session 并读取session

```
<?php session_start(); $_SESSION['name'] = 'Lee'; echo $_SESSION['name']; ?>
```

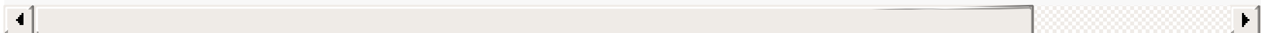
判断session 是否存在

```
<?php session_start(); $_SESSION['name'] = 'Lee'; if (isset($_SESSION['name'])) { echo $_
} ?>
```



删除session

```
<?php session_start(); $_SESSION['name'] = 'Lee'; unset($_SESSION['name']); echo $_SESSIO
```



销毁所有session

```
<?php session_start(); $_SESSION['name'] = 'Lee'; $_SESSION['name2'] = 'Lee'; session_des
```



注：文章出自李炎恢**PHP**视频教程，本文仅限交流使用，不得用于商业用途，否则后果自负。

第十三章 上传文件

学习要点：

1.PHP 上传配置

2.$_FILES 数组

3.PHP 上传函数

虽然大多数人认为Web 只包含网页，但HTTP 协议实际上可以传输任何文件，如office 文档、PDF、可执行文件、AVI、压缩文件及各种其他文件类型。虽然FTP 在历史上一直是向服务器上传文件的标准方式，但通过网页上传文件也逐渐流行起来。

一 · **PHP**上传配置

有一些配置指令可用于精细地调节PHP 的文件上传功能。这些指令用来确定是否启用 PHP 的文件上传、可允许的最大上传文件大小、可允许的最大脚本内存分配和其他各种重要的资源。

1.file_uploads=on|off ： 确定服务器上的PHP 脚本是否可以接受文件上传。

2.max_execution_time=integer ：PHP 脚本在注册一个致命错误之前可以执行的最长时间，以秒为单位。

3.memory_limit=integer ： 设置脚本可以分配到的最大内存，以MB 为单位。这可以防止失控的脚本独占服务器内存。

4.upload_max_filesize=integer ：设置上传文件最大大小，以MB 为单位。此指令必须小于post_max_size。

5.upload_tmp_dir=string ：设置上传文件在处理之前必须存放在服务器的临时一个位置，直到文件移动到最终目的地为止。

6.post_max_size=integer ：确定通过POST 方法可以接受的信息的最大大小，以MB 为单位。

二 · **$_FILES**数组

上传表单的HTML

```
<form enctype="multipart/form-data" action="upload.php" method="post">
<input type="hidden" name="MAX_FILE_SIZE" value="1000000" /> 上传文件: <input type="file" n
<input type="submit" value="上传" />
</form>
```

ENCTYPE="multipart/form-data"，这里是固定写法，否则文件上传失败

ACTION="upload.php“，定义要处理上传的程序文件路径

METHOD="post"，定义传输方式为POST，一般情况下Form提交数据都设置为POST

＜input type="hidden" name="MAX_FILE_SIZE" value="1000000"＞，这是一个隐藏域，定义了上传文件的大小上限，超过这个值时，上传失败。它必须定义在文件上传域的前面.而且这里定义的值不能超过在php.ini 文件中upload_max_filesize 设置的值,否则没有意义了.（注意：MAX_FILE_SIZE 的值只是对浏览器的一个建议，实际上它可以被简单的绕过。因此不要把对浏览器的限制寄希望于该值。实际上，PHP.ini设置中的上传文件最大值，是不会失效的。但是最好还是在表单中加上MAX_FILE_SIZE，因为它可以避免用户在花时间等待上传大文件之后才发现该文件太大了的麻烦。）

<input type="file" name="userfile" />,这是文件上传域,Type属性必须设置为file, 但Name属性可以自定义,这个值会在代码文件中使用.

$_FILES 超级全局变量，它储存各种与上传有关的信息，这些信息对于通过PHP 脚本上传到服务器的文件至关重要。

1.存储在$_FILES["userfile"]["tmp_name"] 变量中的值就是文件在Web 服务器中临时存储的位置。

2.存储在$_FILES["userfile"]["n ame"]变量中的值就是用户系统中的文件名称。

3.存储在$_FILES["userfile"]["s ize"]变量中的值就是文件的字节大小。

4.存储在$_FILES["userfile"]["type"]变量中的值就是文件的MIME 类型，例如：text/plain或image/gif。

5.存储在$_FILES["userfile"]["error"]变量中的值将是任何与文件上载相关的错误代码。这是在PHP4.2.0 中增加的新特性。error 分别提供了一些数组常量：0:表示没有发生错误，1:表示上载文件的大小超出了约定值。文件大小的最大值是PHP 配置文件中指定的，该指令是upload_max_filesize。2:表示上载文件大小超出了HTML 表单的MAX_FILE_SIZE 元素所

指定的最大值。3:表示文件只被部分上载。4:表示没有上载任何文件。

```
<?php print_r($_FILES); ?>
```

三．**PHP**上传函数

PHP 的文件系统库中提供了大量文件处理函数，除此之外，PHP 还提供了两个专门用

于文件上传过程的函数：is_uploaded_file()和move_uploaded_file()。

1.确定是否上传文件：is_uploaded_file()

```
<?php if (is_uploaded_file($_FILES["userfile"]["tmp_name"])) { echo '已经上传到临时文件夹';
} else { echo '失败';
} ?>
```

2.移动上传文件：move_uploaded_file()

```
<?php if (!move_uploaded_file($_FILES["userfile"]["tmp_name"],$_FILES["userfile"]["name"]
} ?>
```

注：文章出自李炎恢**PHP**视频教程，本文仅限交流使用，不得用于商业用途，否则后果自负。

第十四章 处理图像

学习要点：

1.创建图像

2.简单小案例

```
在PHP5 中，动态图象的处理要比以前容易得多。PHP5 在php.ini 文件中包含了GD 扩
```

展包，只需去掉GD 扩展包的相应注释就可以正常使用了。PHP5 包含的GD 库正是升级的GD2 库，其中包含支持真彩图像处理的一些有用的JPG 功能。

```
一般生成的图形，通过PHP 的文档格式存放，但可以通过HTML 的图片插入方式SRC
```

来直接获取动态图形。比如，验证码、水印、微缩图等。

一．创建图像

创建图像的一般流程：

1).设定标头，告诉浏览器你要生成的MIME 类型。

2).创建一个图像区域，以后的操作都将基于此图像区域。

3).在空白图像区域绘制填充背景。

4).在背景上绘制图形轮廓输入文本。

5).输出最终图形。

6).清除所有资源。

7).其他页面调用图像。

设定标头指定MIME 输出类型

```
<?php header('Content-Type: image/png'); ?>
```

创建一个空白的图像区域

```
<?php $im = imagecreatetruecolor(200,200); ?>
```

在空白图像区域绘制填充背景

```
<?php $blue = imagecolorallocate($im,0,102,255);
imagefill($im,0,0,$blue); ?>
```

在背景上绘制图形轮廓输入文本

```
<?php $white = imagecolorallocate($im,255,255,255);
imageline($im,0,0,200,200,$white);
imageline($im,200,0,0,200,$white);
imagestring($im, 5, 80, 20, "Mr.Lee", $white); ?>
```

输出最终图形

```
<?php
imagepng($im); ?>
```

清除所有资源

```
<?php
imagedestroy($im); ?>
```

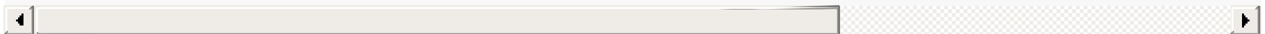
其他页面调用创建的图形

```
<img src="Demo4.php" alt="PHP 创建的图片" />
```

二 · 简单小案例

简单验证码

```
<?php header('Content-type: image/png'); for($Tmpa=0;$Tmpa<4;$Tmpa++){ $nmsg.=dechex(rand
} $im = imagecreatetruecolor(75,25); $blue = imagecolorallocate($im,0,102,255); $white =
imagefill($im,0,0,$blue);
imagestring($im,5,20,4,$nmsg,$white);
imagepng($im);
imagedestroy($im); ?>
```



加载已有的图像

```
<?php header('Content-Type:image/png'); define('__DIR__',dirname(__FILE__).'\\'); $im = i
imagestring($im,3,5,5,'http://www.yc60.com',$white);
imagepng($im);
imagedestroy($im); ?>
```



加载已有的系统字体

```
<?php $text = iconv("gbk","utf-8","李炎恢"); $font = 'C:\WINDOWS\Fonts\SIMHEI.TTF';
imagettftext($im,20,0,30,30,$white,$font,$text); ?>
```

图像微缩

```
<?php header('Content-type: image/png'); define('__DIR__',dirname(__FILE__).'\\'); list($
imagecopyresampled($im2, $im, 0, 0, 0, 0,
$new_width, $new_height, $width, $height);
imagepng($im2);
imagedestroy($im);
Imagedestroy($im2); ?>
```

注：文章出自李炎恢PHP视频教程，本文仅限交流使用，不得用于商业用途，否则后果自负。

第十五章 MySQL 数据库

学习要点：

1.Web 数据库概述

2.MySQL 的操作

3.MySQL 常用函数

4.SQL 语句详解

5.phpMyadmin

一．**Web**数据库概述

现在，我们已经熟悉了PHP 的基础知识，这是我们想暂时离开PHP 一章，来重点介绍

一下关系型数据库，让大家了解数据库比文件储存的有点。这些优点包括：

1.关系型数据库比普通文件的数据访问速度更快。

2.关系型数据库更容易查阅并提取满足特定条件的数据。

3.关系型数据库更具有专门的内置机制处理并发访问，作为程序员，不需要为此担心。

4.关系型数据库可以提供对数据的随即访问。

5.关系型数据库具有内置的权限系统。

关系数据库的概念

至今为止，关系数据库是最常用的数据库类型。在关系代数方面，它们具有很好的理论

基础。当使用关系数据库的时候，并不需要了解关系理论（这是一件好事），但是还是需要

理解一些关于数据库的基本概念。

1）表格

关系数据库由关系组成，这些关系通常称为表格。顾名思义，一个关系就是一个数据的

表格。电子数据表就是一种表格。

编号	姓名	地址	电话
1	周杰伦	台湾高雄	0323839233
2	陈道明	大陆上海	0212324534
3	李炎恢	大陆盐城	0152343434

2）列

表中的每一列都有惟一的名称，包含不同的数据。此外，每一列都有一个相关的数据类型。

3）行

表中的每一行代表一个客户。每一行具有相同的格式，因而也具有相同的属性。行也成为记录。

4）值

每一行由对应每一列的单个值组成。每个值必须与该列定义的数据类型相同。

5）键

每一条数据所对应的唯一的标识。

6）模式

数据库整套表格的完整设计成为数据库的模式。

7）关系

外键标识两个表格数据的关系。

如何设计Web 数据库

1）考虑要建模的实际对象。

2）避免保存冗余数据。

3）使用原子列值（对每一行的每个属性只存储一个数据。）

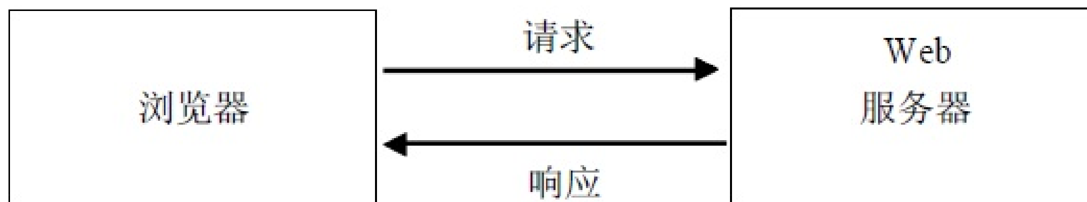
4）选择有意义的键。

5）考虑需要询问数据库的问题。

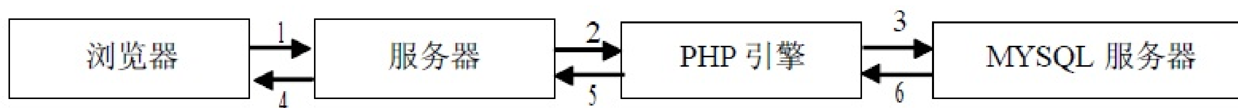
6）避免多个空属性的设计

Web 数据库架构

浏览器和Web 服务器之间的通信：



浏览器和PHP&MySQL 服务器之间的通信



1)用户的Web 浏览器发出HTTP 请求，请求特定Web 页面。

2)Web 服务器收到.php 的请求获取该文件，并将它传到PHP 引擎，要求它处理。

3)PHP 引擎开始解析脚本。脚本中有一条连接数据库的命令，还有执行一个查询的命令。

PHP 打开通向MYSQL 数据库的连接，发送适当的查询。

4)MYSQL 服务器接收数据库查询并处理。将结果返回到PHP 引擎。

5)PHP 以你去哪干完成脚本运行，通常，这包括将查询结果格式化成HTML 格式。然

后再输出HTML 返回到Web 服务器。

6)Web 服务器将HTML 发送到浏览器。

二．**MySQL**操作

登录到MySQL

1)打开MySQL Command Line Client

2)输入root 的设置密码

MySQL 常规命令

1)显示当前数据库的版本号和日期。

SELECT VERSION(),CURRENT_DATE();

2)通过AS 关键字设置字段名。

SELECT VERSION() AS version; //可设置中文，通过单引号

3)通过SELECT 执行返回计算结果

SELECT (20+5)*4;

4)通过多行实现数据库的使用者和日期

>SELECT

>USER()

>,

>NOW()

>;

5)通过一行显示数据库使用者和日期

>SELECT USER();SELECT NOW();

6)命令的取消

>\c

7)MySQL 窗口的退出

>exit;

MySQL 常用数据类型

整数型：TINYINT，SMALLINT，INT，BIGINT

浮点型：FLOAT，DOUBLE，DECIMAL(M,D)

字符型：CHAR，VARCHAR

日期型：DATETIME，DATE，TIMESTAMP

备注型：TINYTEXT，TEXT，LONGTEXT

日期型

列类型	“零”值
DATETIME	'0000-00-00 00:00:00'
DATE	'0000-00-00'
TIMESTAMP	00000000000000
TIME	'00:00:00'
YEAR	0000

字符串型

值	CHAR(4)	存储需求	VARCHAR(4)	存储需求
''	' '	4个字节	''	1个字节
'ab'	'ab '	4个字节	'ab '	3个字节
'abcd'	'abcd'	4个字节	'abcd'	5个字节
'abcdefgh'	'abcd'	4个字节	'abcd'	5个字节

整数型

类型	字节	最小值	最大值
		(带符号的/无符号的)	(带符号的/无符号的)
TINYINT	1	-128	127
		0	255
SMALLINT	2	-32768	32767
		0	65535
MEDIUMINT	3	-8388608	8388607
		0	16777215
INT	4	-2147483648	2147483647
		0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807
		0	18446744073709551615

整数型

类型	字节	最小值	最大值
FLOAT	4	+-1.175494351E-38	+-3.402823466E+38
DOUBLE	8	+-2.2250738585072014E-308	+-1.7976931348623157E+308
DECIMAL	可变	它的取值范围可变。	

备注型

类型	描述
TINYTEXT	字符串，最大长度255个字符
TEXT	字符串，最大长度65535个字符
MEDIUMTEXT	字符串，最大长度16777215个字符
LONGTEXT	字符串，最大长度4294967295个字符

MySQL 数据库操作

1)显示当前存在的数据库

>SHOW DATABASES;

2)选择你所需要的数据库

>USE guest;

3)查看当前所选择的数据库

>SELECT DATABASE();

4)查看一张表的所有内容

>SELECT * FROM guest; //可以先通过SHOW TABLES;来查看有多少张表

5)根据数据库设置中文编码

>SET NAMES gbk; //set names utf8;

6)创建一个数据库

>CREATE DATABASE book;

7)在数据库里创建一张表

>CREATE TABLE users (

>username VARCHAR(20), //NOT NULL 设置不允许为空

>sex CHAR(1),

>birth DATETIME);

8)显示表的结构

>DESCIRBE users;

9)给表插入一条数据

>INSERT INTO users (username,sex,birth) VALUES ('Lee','x',NOW());

10)筛选指定的数据

> SELECT * FROM users WHERE username = 'Lee';

11)修改指定的数据

>UPDATE users SET sex = '男' WHERE username='Lee';

12)删除指定的数据

> DELETE FROM users WHERE username='Lee';

13)按指定的数据排序

> SELECT * FROM users ORDER BY birth DESC; //正序

14)删除指定的表

>DROP TABLE users;

15)删除指定的数据库

>DROP DATABASE book;

三 · **MySQL**常用函数

文本函数

函数	用法	描述
CONCAT()	CONCAT(x,y,...)	创建形如 xy 的新字符串
LENGTH()	LENGTH(column)	返回列中储存的值的长度
LEFT()	LEFT(column,x)	从列的值中返回最左边的 x 个字符
RIGHT()	RIGHT(column,x)	从列的值中返回最右边的 x 个字符
TRIM()	TRIM(column)	从存储的值删除开头和结尾的空格
UPPER()	UPPER(column)	把存储的字符串全部大写
LOWER()	LOWER(column)	把存储的字符串全部小写
SUBSTRING()	SUBSTRING(column, start, length)	从 column 中返回开始 start 的 length 个字符（索引从 0 开始）
MD5()	MD5(column)	把储存的字符串用 MD5 加密
SHA()	SHA(column)	把存储的字符串用 SHA 加密

数字函数

函数	用法	描述
ABS()	ABS(x)	返回 x 的绝对值
CEILING()	CEILING(x)	返回 x 的值的最大整数
FLOOR()	FLOOR(x)	返回 x 的整数
ROUND()	ROUND(x)	返回 x 的四舍五入整数
MOD()	MOD(x)	返回 x 的余数
RNAD()	RNAD()	返回 0-1.0 之间随机数
FORMAT()	FORMAT(x,y)	返回一个格式化后的小数
SIGN()	SIGN(x)	返回一个值，正数(+1)，0，负数 (-1)
SQRT()	SQRT(x)	返回 x 的平方根

日期和时间函数

函数	用法	描述
HOUR()	HOUR(column)	只返回储存日期的小时值
MINUTE()	MINUTE(column)	只返回储存日期的分钟值
SECOND()	SECOND(column)	只返回储存日期的秒值
DAYNAME()	DAYNAME(column)	返回日期值中天的名称
DAYOFMONTH()	DAYOFMONTH(column)	返回日期值中当月第几天
MONTHNAME()	MONTHNAME(column)	返回日期值中月份的名称
MONTH()	MONTH(column)	返回日期值中月份的数字值
YEAR()	YEAR(column)	返回日期值中年份的数字值
CURDATE()	CURDATE()	返回当前日期
CURTIME()	CURTIME()	返回当前时间
NOW()	NOW()	返回当前时间和日期

格式化日期和时间(DATE_FORMAT()和 TIME_FORMAT())

名词	用法	示例
%e	一月中的某天	1~31
%d	一月中的某天，两位	01~31
%D	带后缀的天	1st~31st
%W	周日名称	Sunday~Saturday
%a	简写的周日名称	Sun-Sat
%c	月份编号	1~12
%m	月份编号，两位	01~12
%M	月份名称	January~December
%b	简写的月份名称	Jan~Dec
%Y	年份	2002
%y	年份，两位	02
%l	小时	1~12
%h	小时,两位	01~12
%k	小时，24 小时制	0~23
%H	小时，24 小制度，两位	00~23
%i	分钟	00~59
%S	秒	00~59
%r	时间	8:17:02 PM
%T	时间，24 小时制	20:17:02 PM
%p	上午或下午	AM 或 PM

四．**SQL**语句详解

1.创建一个班级数据库school，里面包含一张班级表grade，包含编号(id)、姓名(name)、邮件(email)、评分(point)、注册日期(regdate)。

mysql>CREATE DATABASE school; //创建一个数据库

mysql> CREATE TABLE grade (

//UNSIGNED 表示无符号，TINYINT(2) 无符号整数0-99，NOT NULL 表示不能为

空，AUTO_INCREMENT 表示从1 开始没增加一个字段，累计一位

-> id TINYINT(2) UNSIGNED NOT NULL AUTO_INCREMENT,

-> name VARCHAR(20) NOT NULL,

-> email VARCHAR(40),

-> point TINYINT(3) UNSIGNED NOT NULL,

-> regdate DATETIME NOT NULL,

-> PRIMARY KEY (id) //表示id 为主键，让id 值唯一，不得重复。

->);

2.给这个班级表grade 新增5-10 条学员记录

mysql> INSERT INTO grade (name,email,point,regdate) VALUES

('Lee','yc60.com@gmail.com',95,NOW());

3.查看班级所有字段的记录，查看班级id,name,email 的记录

mysql> SELECT * FROM grade;

mysql> SELECT id,name,email FROM grade;

WHERE 表达式的常用运算符

MYSQL 运算符	含义
=	等于
<	小于
>	大于
<=	小于或等于
>=	大于或等于
!=	不等于
IS NOT NULL	具有一个值
IS NULL	没有值
BETWEEN	在范围内
NOT BETWEEN	不在范围内
IN	指定的范围
OR	两个条件语句之一为真
AND	两个条件语句都为真
NOT	条件语句不为真

4.姓名等于'Lee'的学员，成绩大于90 分的学员，邮件不为空的成员，70-90 之间的成员

mysql> SELECT * FROM grade WHERE name='Lee';

mysql> SELECT * FROM grade WHERE point>90;

mysql> SELECT * FROM grade WHERE email IS NOT NULL;

mysql> SELECT * FROM grade WHERE point BETWEEN 70 AND 90;

mysql> SELECT * FROM grade WHERE point IN (95,82,78);

5.查找邮件使用163 的学员，不包含yc60.com 字符串的学员

mysql> SELECT * FROM grade WHERE email LIKE '%163.com';

mysql> SELECT * FROM grade WHERE email NOT LIKE '%yc60.com%';

6.按照学员注册日期的倒序排序，按照分数的正序排序

mysql> SELECT * FROM grade ORDER BY regdate DESC;

mysql> SELECT * FROM grade ORDER BY point ASC;

7.只显示前三条学员的数据，从第3 条数据开始显示2 条

mysql> SELECT * FROM grade LIMIT 3;

mysql> SELECT * FROM grade LIMIT 2,2;

8.修改姓名为'Lee'的电子邮件

mysql> UPDATE grade SET email='yc60.com@163.com' WHERE name='Lee';

9.删除编号为4 的学员数据

mysql> DELETE FROM grade WHERE id=4;

MYSQL 分组函数

函数	用法	描述
AVG()	AVG(column)	返回列的平均值
COUNT()	COUNT(column)	统计行数
MAX()	MAX(column)	求列中的最大值
MIN()	MIN(column)	求列中的最小值
SUM()	SUM(column)	求列中的和

10.过一遍以上的分组函数

略。

11.检查这个表的信息

mysql> SHOW TABLE STATUS \G;

12.优化一张表

mysql> OPTIMIZE TABLE grade;

五·**PhpMyAdmin**

phpMyAdmin(简称PMA)是一个用PHP 编写的，可以通过互联网在线控制和操作MySQL。他是众多MySQL 管理员和网站管理员的首选数据库维护工具，通过phpMyAdmin可以完全对MySQL 数据库进行操作。

创建数据库scholl

创建一个数据库->选择utf8 字符集

导出另一个数据库SQL

1.选择另一个数据库->导出

2.选择需要导出的表->全选

3.选择Add DROP TABLE / DROP VIEW （基本表一旦删除，表中的数据以及相应建立的索引和视图都将自动被删除）

4.选择另存为文件

5.选择执行，保存sql 文件

导入数据库

1.选择被导入的数据库

2.选择Import(导入)，选择sql 文件

3.执行即可

删除表

1.可以直接选择操作中的✕，然后确认即可删除数据表.

2.也可以选择复选按钮☐.然后选择选中项：[选中项:]，选择删除，执行即可

重建表

1.找到sql 文件中的刚才输出的建表语句.

2.复制建表语句

3.然后选择sql，选择粘贴，执行即可

修复数据表

1.选择要修复的表

2.在选中项中，选择修复表,即可

优化数据表

1.选择要优化的表

2.在选中项中，选择优化表，即可

修改，删除，插入表记录

执行SQL 语句

注：文章出自李炎恢**PHP**视频教程，本文仅限交流使用，不得用于商业用途，否则后果自负。

第十六章 PHP 操作MySQL

学习要点：

1.PHP 连接到MySQL

2.增删改查

3.其他常用函数

如果你已经具有了使用PHP、SQL 和MySQL 的丰富经验，现在就可以把所有这些技术组合在一起。PHP 与MySQL 之间稳固的集成只是众多程序员采纳它的一个原因，还有一个原因就是，它如此的简单方便。

一．PHP连接到MySQL

这里，我们全面采用UTF-8 编码。

设置Zend Stduio 的编码：Window -> Preferences -> Workspace

标头设置，让火狐和IE 保持编码统一：

```
<?php header('Content-Type:text/html; charset=utf-8'); ?>
```

连接MySQL

```
<?php $conn = @mysql_connect(DB_HOST,DB_USER,DB_PASSWORD) or die('数据库连接失败！错误信息：'
```

数据库连接参数，可以用常量存储，这样就不能修改，更加安全。

```
<?php define('DB_USER','root'); define('DB_PASSWORD','yangfan'); define('DB_HOST','localh
```

选择你所需要的数据库

```
<?php
@mysql_select_db(DB_NAME) or die('数据库找不到！错误信息：'.mysql_error()); ?>
```

设置字符集，如果是GBK，直接设置SET NAMES GBK 即可

```
<?php
@mysql_query('SET NAMES UTF8') or die('字符集设置错误'); ?>
```

获取记录集

```
<?php $query = "SELECT * FROM grade"; $result = @mysql_query($query) or die('SQL 语句有误！
```

输出一条记录

```
<?php print_r(mysql_fetch_array($result,MYSQL_ASSOC)); ?>
```

释放结果集资源

```
<?php mysql_free_result($result); ?>
```

关闭数据库

```
<?php mysql_close($conn); ?>
```

二．增删改查

新增数据

```
<?php $query = "INSERT INTO grade (name,email,point,regdate) VALUE
('李炎恢','yc60.com@gmail.com',,NOW())";
@mysql_query($query) or die('添加数据出错：'.mysql_error()); ?>
```

修改数据

```
<?php $query = "UPDATE grade SET name='小可爱' WHERE id=6";
@mysql_query($query) or die('修改出错：'.mysql_error()); ?>
```

删除数据

```
<?php $query = "DELETE FROM grade WHERE id=6";
@mysql_query($query) or die('删除错误：'.mysql_error()); ?>
```

显示数据

```
<?php $query = "SELECT id,name,email,point FROM grade"; $result = @mysql_query($query) or
} ?>
```

三．其他常用函数

mysql_fetch_row()：从结果集中取得一行作为枚举数组

mysql_fetch_assoc()： 从结果集中取得一行作为关联数组

mysql_fetch_array()： 从结果集中取得一行作为关联数组，或数字数组，或二者兼有

mysql_fetch_lengths ()： 取得结果集中每个输出的长度

mysql_field_name()： 取得结果中指定字段的字段名

mysql_num_rows()： 取得结果集中行的数目

mysql_num_fields()：取得结果集中字段的数目

mysql_get_client_info()： 取得MySQL 客户端信息

mysql_get_host_info()： 取得MySQL 主机信息

mysql_get_proto_info()： 取得MySQL 协议信息

mysql_get_server_info()： 取得MySQL 服务器信息

注：文章出自李炎恢**PHP**视频教程，本文仅限交流使用，不得用于商业用途，否则后果自负。

第十七章 面向对象基础

学习要点：

1.什么是面向对象

2.OOP 的特点

3.关键的OOP 概念

4.创建OOP

许多语言本身就是面向对象（OOP）的，而PHP 用了几年才引入了这类功能。面向对象的诞生是开发范型一次的重大改变，编程的注意力重新从应用程序的逻辑回到其数据上来。换句话说，OOP 将焦点从编程的过程性事件转向最终建模的眞实实体。这使得应用程序更接近于我们周围的现实世界。

一．什么是面向对象

面向过程

这就好比你是公司的一名员工，今天有个任务要在公司组装一批电脑。那么你就开始采购、讨价还价、运输回公司、开始组装、布线网络、调试机器、完成。也就是说，面向过程就是具体化的实现，细节明确。

面向对象

这就好像你是公司的总裁，你布置给一名员工一个组装一批电脑的任务。完毕。也就是说，面向对象就是抽象化的执行，具体还是由那名员工来完成。而细节方面，总裁不需要去考虑。这样的好处是显而易见的，在有管理高层的公司可以协调作业，而没有管理高层的公司，只有一些具体实现功能的员工，会乱做一团。

二．**OOP**的特点

封装

隐藏对象的属性和实现细节，仅对外公开接口,控制在程序中属性的读和修改的访问级别；将抽象得到的数据和行为（或功能）相结合，形成一个有机的整体，也就是将数据与操作数据的源代码进行有机的结合，形成“类”，其中数据和函数都是类的成员。

继承

继承是从一个基类得到一个或多个类的机制。

继承自另一个类的类被称为该类的子类。这种关系通常用父亲和孩子来比喻。子类将继承父类的特性。这些特性由属性和方法组成。子类可以增加父类之外的新功能，因此子类也被称为父类的“扩展”。

多态

多态是指OOP 能够根据使用类的上下文来重新定义或改变类的性质或行为，或者说接口的多种不同的实现方式即为多态。把不同的子类对象都当作父类来看，可以屏蔽不同子类对象之间的差异，写出通用的代码，做出通用的编程，以适应需求的不断变化。

三 · 关键的 **OOP**概念

类（**class**）

类是对某个对象的定义。它包含有关对象动作方式的信息，包括它的名称、方法、属性和事件。实际上它本身并不是对象，因为它不存在于内存中。当引用类的代码运行时，类的一个新的实例，即对象，就在内存中创建了。虽然只有一个类，但能从这个类在内存中创建多个相同类型的对象。

对象（**object**）

对象是一件事、一个实体、一个名词，可以获得的东西，可以想象有自己的标识的任何东西。对象是类的实例化。一些对象是活的，一些对象不是。

比如这辆汽车、这个人、这间房子、这张桌子、这株植物、这张支票、这件雨衣。概括来说就是：一切皆对象。

例如：类是对象的抽象定义，说白了，如果这个对象是电脑，类可以创建出许多对象，类可以生成很多电脑，再白一点，类可以当成一个电脑生产厂，可以生成出很多很多台电脑。

字段（**filed**）

字段是用于描述类的某方面的性质，它与一般的PHP 变量非常相似，只是有一些细微的差别。

例如：电脑品牌，电脑的型号等特性。

属性（attribute）

通过方法来访问和操作字段，一方面可以保护字段，同时还允许访问公共字段一样访问数据。

例如：获取电脑品牌，设置电脑品牌等操作。

方法（method）

方法与函数非常相似，只不过方法是用来定义类的行为。与函数一样，方法可以接受输入参数，可以向调用者返回一个值。

例如：打开电脑，输入文本，运行程序。

四 · 创建 OOP

类的创建：

```
class Computer { //类的字段(成员)
//类的方法
}
```

对象的声明：

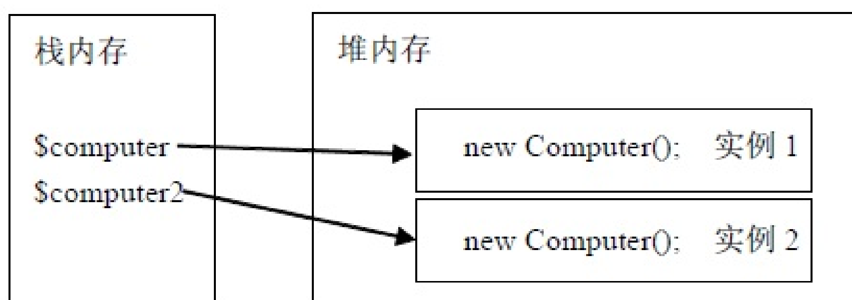
```
$computer = new Computer();
```

new 标识符是为了在内测中创建一个对象（实例），而Computer()就是那个类所生成的实例。

$computer 是一个变量，而且又是生成实例的引用。

有时，你可能需要创建多个对象。

```
$computer2 = new Computer();
```



使用var_dump()函数可以打印变量的相关信息。

字段（成员）的添加：

```
class Computer {
    //类的字段(成员)
    public $_name = '联想120';
    public $_model = 'LX';
}
```

1.声明字段：

```
public $_name = '联想120'
```

a) public 是修饰符，表示这是一个公共字段，可以通过外部直接访问。

b) $_name 是变量名。

c) '联想120'是变量的值。

2.读取字段

```
echo $computer->_name;
```

3.设置字段

```
$computer->_name = 'IBM110';
```

方法的创建

```
class Computer { //类的方法
     function run() { echo '我成功的运行了！';
     }
}
```

执行方法

```
$computer->run();
```

也可以在方法添加一些参数，执行的时候，传入这些参数。

```
class Computer { //类的方法
     function run($_what) { echo $_what . '成功的运行了！';
     }
} $computer = new Computer (); $computer->run('电脑');
```

构造方法

所谓构造方法，也是方法，只不过是一种特殊的方法。而方法名必须和类名一致，

并且不需要像普通方法一样，必须通过调用才能执行，只需要实例化即完成调用过程。

一般来说，构造方法总是在做一些初始化的工作。

```
class Computer { //构造方法
    function Computer() { echo '我是构造方法！';
    }
} new Computer (); //这样即完成了调用
```

在PHP5 我们可以通过__construct 的内置方法来识别构造方法，而不用再需要和

类名相同了。

```
class Computer { //构造方法
    function __construct() { echo '我是构造方法！';
    }
}
```

相对应构造方法，还有一种内置的方法是析构方法，它的用途在整个类使用完毕都

执行。一般可用于清理内存中对象(脚本执行完毕的时候会自动清理)。而如果有脚本执

行完毕后并没有清理的，比如数据库数据等，就有必要使用析构方法。

```
class Computer { //析构方法
    function __destruct() { echo '我是析构方法';
    }
}
```

注：文章出自李炎恢**PHP**视频教程，本文仅限交流使用，不得用于商业用途，否则后果自负。

第十八章 面向对象的特性

学习要点：

1.OOP 的封装

2.OOP 的继承

3.OOP 的多态

面向对象的三个主要特性是封装、继承和多态。

一．**OOP**的封装

隐藏对象的字段和实现细节，仅对外公开接口,控制在程序中字段的读和修改的访问级别；将抽象得到的数据和行为（或功能）相结合，形成一个有机的整体，也就是将数据与操作数据的源代码进行有机的结合，形成“类”，其中数据和函数都是类的成员。

字段的作用域

1.public 公共的(类外可以访问)

2.private 私有的(类内可以访问)

3.protected 受保护的(类内和子类可以访问，类外不可访问)

创建使用了私有的字段，这样外部就无法访问了

```
class Computer { //类的字段(成员)
    private $_name = '联想120'; private $_model = 'LX';
}
```

通过一个公共方法作为入口，访问私有字段，而必须使用$this关键字。

```
class Computer { //类的字段(成员)
    private $_name = '联想120'; private $_model = 'LX'; //通过公共方法来访问私有字段
    function run() { echo $this->_name;
    }
} $computer->run ();
```

属性操作**(**私有字段的赋值与取值**)**

可以设计两个公共方法，一个方法为setName()，用于赋值；一个方法为getName()，用于取值。

```
class Computer { //类的字段(成员)
    private $_name; private $_model; //赋值
    function setName($_name) { $this->_name = $_name;
    } //取值
    function getName() { return $this->_name;
    }
} $computer = new Computer (); $computer->setName ( 'IBM' ); echo $computer->getName ();
```

如果有十个字段那么就必须要二十个方法才能够赋值和取值，那么有没有更简便的方法呢？PHP内置两个方法(拦截器)专门用于取值与赋值：**set()**，get()。

```
class Computer { //类的字段(成员)
    private $_name; private $_model; //所有字段的赋值都在这里进行
    function __set($_key, $_value) { $this->$_key = $_value;
    } //所有字段的取值都在这里进行
    function __get($_key) { return $this->$_key;
    }
} $computer = new Computer (); $computer->_model = 'LX'; echo $computer->_model;
```

方法私有：有些使用类里面的方法并不需要对外公开，只是里面运作的一部分，这个时候可以将方法也封装起来。

```
class Computer { //类的字段(成员)
    private $_name; private $_model; //私有方法
    private function getEcho() { echo '我是私有化的方法';
    } //公共方法一般是对外的入口
    public function run() { $this->getEcho ();
    }
} $computer = new Computer (); $computer->run ();
```

建议：方法前面如果没有修饰符，那么就是外部可访问的公共方法，但为了让程序更加的清晰，建议在前面加上public。

常量（**constant**）

在类中可以定义常量，用来表示不会改变的值。对于从该类实例化的任何对象来说，常量值在这些对象的整个生命周期中都保持不变。

```
class Computer { const PI = 3.1415926;
} echo Computer::PI;
```

静态类成员

有时候，可能需要创建供所有类实例共享的字段和方法，这些字段和方法与所有的类实例有关，但不能由任何特定对象调用。

```
class Computer { public static $_count = 0;
} echo Computer::$_count;
```

一般来说，必须将字段做成私有化。所以可能需要这么做：

```
class Computer { private static $_count = 0; public static function setRun() {
        self::$_count ++;
    } public static function getRun() { return self::$_count;
    }
}
Computer::setRun (); echo Computer::getRun ();
```

Instanceof关键字

PHP5有一个instanceof关键字，使用这个关键字可以确定一个对象是类的实例、类的

子类，还是实现了某个特定接口，并进行相应的操作。

```
class Computer {

} $computer = new Computer (); echo ($computer instanceof Computer);
```

二．OOP继承

继承是从一个基类得到一个或多个类的机制。

继承自另一个类的类被称为该类的子类。这种关系通常用父类和孩子来比喻。子类将继

承父类的特性。这些特性由属性和方法组成。子类可以增加父类之外的新功能，因此子类也

被称为父类的“扩展”。

在PHP中，类继承通过extends关键字实现。继承自其他类的类成为子类或派生类，子

类所继承的类成为父类或基类。(PHP只支持单继承，PHP不支持方法重载)。

```
class Computer { private $_name = '联想120'; private function __get($_key) { return $this-
    } public function run() { echo '我是父类';
    }
} class NoteBookComputer extends Computer {
} $notebookcomputer = new NoteBookComputer (); $notebookcomputer->run (); echo $notebookc
```

字段和方法的重写（覆盖）

有些时候，并不是特别需要父类的字段和方法，那么可以通过子类的重写来修改父类的

字段和方法。

```
class Computer { public $_name = '联想120'; protected function run() { echo '我是父类';
    }
} class NoteBookComputer extends Computer { public $_name = 'IBM'; public function run()
    }
}
```

子类调用父类的字段或方法

为了安全，我们一般将父类的方法封装了起来，这样，外部就无法调用，只能被继承它

的子类所看到。这个时候，就需要通过子类操作来调用父类了。

```
class Computer { protected $_name = '联想120'; protected function run() { echo '我是父类';
    }
} class NoteBookComputer extends Computer { public function getName() { echo $this->_name
    } public function getRun() { echo $this->run ();
    }
}
```

通过重写调用父类的方法

有的时候，我们需要通过重写的方法里能够调用父类的方法内容，这个时候就必须使用

语法：父类名::方法() 或者parent::方法()即可调用。

```
class Computer { protected function run() { echo '我是父类';
    }
} class NoteBookComputer extends Computer { public function run() { echo Computer::run ()
    }
}
```

final关键字可以防止类被继承，有些时候只想做个独立的类，不想被其他类继承使用，

那么就必须使用这个关键字。建议只要是单独的类，都加上这个关键字。

```
final class Computer { //无法继承的类
    final public function run() {
    } //无法被继承的方法
} class NoteBookComputer extends Computer { //会报错
}
```

抽象类和方法（abstract）

抽象方法很特殊，只在父类中声明，但在子类中实现。只有声明为abstract的类可以声

明抽象方法。

规则：

1.抽象类不能被实例化，只能被继承。

2.抽象方法必须被子类方法重写。

```
abstract class Computer { abstract function run();
} final class NotebookComputer extends Computer { public function run() { echo '我实现了';
    }
}
```

接口（interface）

接口定义了实现某种服务的一般规范，声明了所需的函数和常量，但不指定如何实现。

之所以不给出实现的细节，是因为不同的实体可能需要用不同的方式来实现公共的方法定义。关键是要建立必须实现的一组一般原则，只要满足了这些原则才能说实现了这个接口。

规则：

1.类全部为抽象方法（不需要声明abstract）

2.接口抽象方法必须是public

3.成员（字段）必须是常量

```
interface Computer { const NAME = '联想120'; public function run();
} final class NotebookComputer implements Computer { public function run() { echo '实现了
    }
} $notebookcomputer = new NoteBookComputer (); $notebookcomputer->run (); echo Computer::
```

子类可以实现多个接口

```
interface Computer { const NAME = '联想120'; public function run();
} interface Notebook { public function book();
} final class NotebookComputer implements Computer, Notebook { public function run() { ec
    } public function book() { echo '实现了接口的方法';
    }
}
```

三．多态

多态是指OOP 能够根据使用类的上下文来重新定义或改变类的性质或行为，或者说接口的多种不同的实现方式即为多态。把不同的子类对象都当作父类来看，可以屏蔽不同子类对象之间的差异，写出通用的代码，做出通用的编程，以适应需求的不断变化。

```
interface Computer { public function version(); public function work();
} class NotebookComputer implements Computer { public function version() { echo '联想120';
    } public function work() { echo '笔记本正在随时携带运行！';
    }
} class desktopComputer implements Computer { public function version() { echo 'IBM';
    } public function work() { echo '台式电脑正在工作站运行！';
    }
} class Person { public function run($type) { $type->version (); $type->work ();
    }
} $person = new Person (); $desktopcomputer = new desktopComputer (); $notebookcomputer =
```

注：文章出自李炎恢**PHP**视频教程，本文仅限交流使用，不得用于商业用途，否则后果自负。

第十九章 面向对象的工具

学习要点：

1.OOP 的魔术方法

2.类函数和对象函数

3.OOP 的反射API

PHP通过类和方法等语言结构支持面向对象编程。同时也通过对象相关的函数和内置类为面向对象提供广泛支持。

一．**OOP**的魔术方法

PHP引入了**autoload()**内置方法来自动包含类文件。autoload()应该被写成单个参数的方法。当PHP引擎遇到试图实例化未知类的操作时，会调用__autoload()方法，并将类名当作字符串参数传递给它。

```
function __autoload($_className) { require $_className . '.class.php';
} $demo = new Computer ();
```

PHP采用了__call()内置方法来屏蔽对象调用方法时产生的错误。当对象调用一个不存在的方法时，会自动调用__call()方法。

```
private function __call($_methodName,$args) { echo $_methodName.'方法不存在'; print_r($args
} $computer->go('我',1,'知道');
```

PHP使用**toString()**内置方法来打印对象的引用。没有使用toString()的对象是产生一个错误，当打印对象的时候会自动调用__toString()方法。

```
class Computer { private function __toString() { return '打印对象';
    }
} echo new Computer ();
```

PHP可以在类中定义一个__clone()内置方法来调整对象的克隆行为。当一个对象被克隆的时候自动执行__clone()方法，而复制的对象可以在其方法体内进行调整。

```
class Computer { public $_name; public function __clone() { $this->_name = 'ibm';
    }
} $computer1 = new Computer (); $computer1->_name = 'dell'; $computer2 = clone $computer1
```

二·类函数和对象函数

PHP提供了一系列强大的函数来检测类和对象。以便在第三方系统，运行时知道正在使用的是哪个。

1.class_exists()函数接受表示类的字符串，检查并返回布尔值。如果类存在，返回true，否则返回false。

```
echo class_exists('Computer');
```

2.get_class()函数获取对象的类名，如果不是对象，则返回false。

```
echo get_class($computer);
```

3.get_class_methods()函数获取类中的方法(公共的)，以数组的形式返回出来。

```
print_r(get_class_methods($computer));
```

4.get_class_vars()函数获取类中的字段(公共的)，以数组的形式返回出来

```
print_r(get_class_vars('Computer'));
```

5.get_parent_class()函数获取子类的父类，如果没有返回false。

```
echo get_parent_class('NoteComputer');
```

6.interface_exists()函数确定接口是否存在，如果存在返回true，否则返回false。

```
echo interface_exists('Computer');
```

7.is_a()函数确定对象是否是类或者是否是这类的父类时，返回ture，否则返回false。

```
echo is_a($computer,'Computer');
```

8.is_subclass_of()函数确定对象是否是类的子类，是返回ture，否则返回false。

```
echo is_subclass_of($notecomputer,'Computer');
```

9.method_exists()函数确定对象的方法是否存在，是返回ture，否则返回false。

```
echo method_exists($computer,'_run');
```

三．**OOP**的反射**API**

PHP5的类和对象函数并没有告诉我们类内部的所有一切，而只是报告了它们的公共成员。要充分了解一个类，需要知道其私有成员和保护成员，还要知道其方法所期望的参数。对此，使用反射API。

1.获得反射API的转储信息

```
$rc = new ReflectionClass('Computer');
Reflection::export($rc);
```

2.获得PHP内置的类库的信息

```
Reflection::export(new ReflectionClass('Reflection'));
```

3.获取类里的某个元素

```
$_rc = new ReflectionClass('Computer'); echo $_rc->getFileName(); echo $_rc->getName();
```

注：文章出自李炎恢**PHP**视频教程，本文仅限交流使用，不得用于商业用途，否则后果自负。